DOCUMENT RESUME

ED 216 683 IR 010 160

| | |
|---|---|
| AUTHOR | Edwards, Judith B.; And Others |
| TITLE | Elements of Computer Careers. |
| INSTITUTION | Northwest Regional Educational Lab., Portland, Oreg. |
| SPONS AGENCY | National Inst. of Education (DHEW), Washington, D.C. |
| REPORT NO | ISBN-0-13-263483-X |
| PUB DATE | 77 |
| CONTRACT | OEG-0-71-4655/NIH-785503 |
| NOTE | 351p.; This item appears as a Level III as ED 139 380. |
| EDRS PRICE | MF01/PC15 Plus Postage. |
| DESCRIPTORS | Career Awareness; *Career Education; Career Exploration; *Computer Literacy; *Computer Oriented Programs; *Computer Science Education; *Data Processing; Input Output Devices; *Occupational Information; Secondary Education; Technical Education; Textbooks |

ABSTRACT

This textbook is intended to provide students with an awareness of the possible alternatives in the computer field and with the background information necessary for them to evaluate those alternatives intelligently. Problem solving and simulated work experiences are emphasized as students are familiarized with the functions and limitations of computers, how information is processed, hardware and software, techniques and tools, and systems concepts. The final chapter provides more specific discussions of several computer-related careers: data preparation clerk (or data clerk), computer operator, computer programmer, systems programmer, systems analyst, and computer center manager. Topics addressed include relative salary levels, requirements of the job (skills, education, personal characteristics, and working conditions), and career potential. The appendix includes an explanation of the use of binary codes and a glossary of terms is included. (CHC)

# ELEMENTS OF COMPUTER CAREERS

Northwest
Regional
Educational
Laboratory

*Principal Author:* Judith B. Edwards

*Editor:* Antoinette Ellis

*Contributing Authors:* Winston Addis
Kathryn Curren
John R. Lynch, Sr.

This publication was prepared under contract number OEG 0-71-4655/NIH-785503 National Institute of Education by the Computer Technology Program, Northwest Regional Educational Laboratory, Duane Richardson Program Director.

Prentice-Hall, Inc., Englewood Cliffs, New Jersey

ELEMENTS OF COMPUTER CAREERS

Northwest Regional Educational Laboratory

Supplementary Materials:
Teacher's Guide
'tudent Guide

This work was developed under a contract with the National Institute of Education, U.S. Department of Health, Education and Welfare. However, the content does not necessarily reflect the position or policy of that Agency, and no official endorsement of these materials should be inferred.

Printed in the United States of America.
10 9 8 7 6 5 4 3 2 1

Prentice-Hall International, Inc., London
Prentice-Hall of Australia, Pty. Ltd., Sydney
Prentice-Hall of Canada, Ltd., Toronto
Prentice-Hall of India Private Ltd., New Delhi
Prentice-Hall of Japan, Inc., Tokyo

ISBN 0-13-263483-X

# PREFACE

In 1971 the U.S. Office of Education entered into a major commitment to the concept of "career education." Fundamental to this concept is the notion that the entire range of educational experience should involve preparation for economic independence and personal fulfillment, and should develop an appreciation for the dignity of work. State educational agencies and local school districts were receptive to this notion, and many state legislatures have mandated career education for all students. ELEMENTS OF COMPUTER CAREERS was developed at Northwest Regional Educational Laboratory, with support from the USOE, to provide awareness, exploration, and hands-on work experience in an occupational area of major importance in our technological world.

The development work was carried out on-site at a Gresham, Oregon, school in a trailer equipped with a complete, operating minicomputer center. Students used the materials as they were developed and took complete responsibility for operating the computer center to provide computer services for teachers and administrators. This use of the materials functioned as formative testing and resulted in much on-site revision, retesting, and more revision.

Field tests were subsequently conducted in four schools, representing a variety of settings, a range of access to computer hardware, and courses of various lengths. The materials were also used to conduct a teacher-training course in Agana, Guam. On the basis of the data from the field tests, the materials were again revised, this time for publication.

ELEMENTS OF COMPUTER CAREERS has maximum flexibility for use. It may be used in a setting where a computer is available, one in which there is access to a remote terminal, or one in which neither is at hand. Through problem solving, simulation, and hands-on activities, the reader is exposed to the working world of computers and has direct experience of the many computer-related careers.

ELEMENTS OF COMPUTER CAREERS has been demonstrated to provide an effective and motivating course of instruction for students and teachers alike, for all levels of instructional objectives from career awareness and computer literacy through career preparation and teacher training.

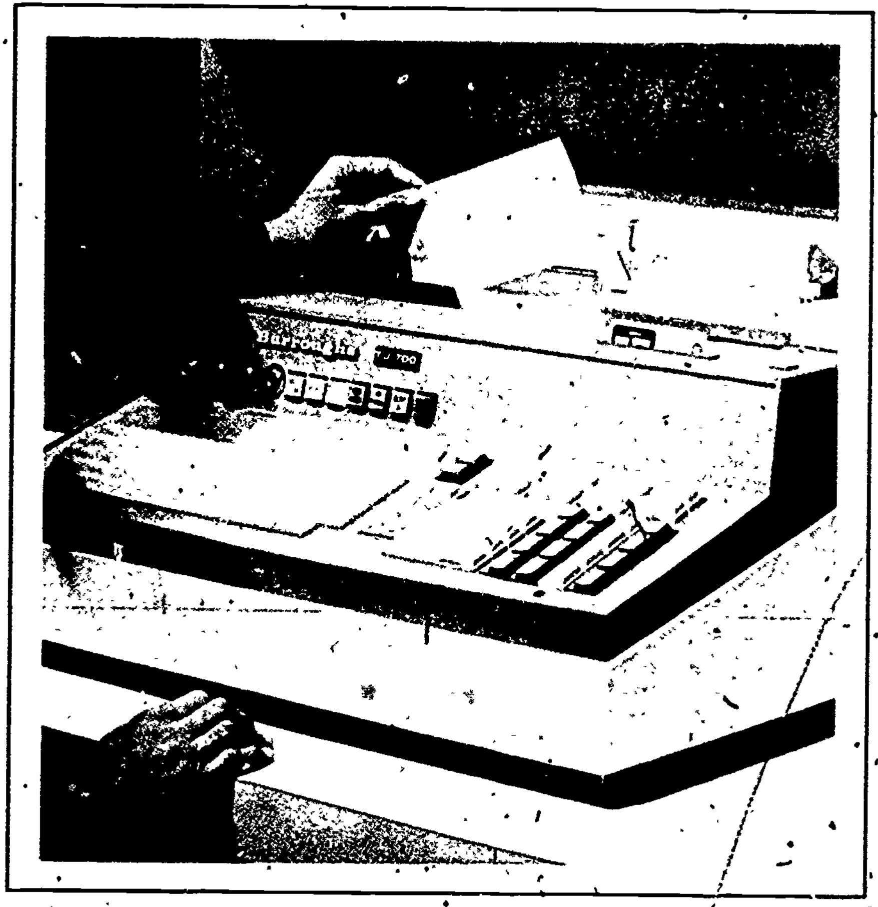
## ILLUSTRATION CREDITS

The photographs in this book appear courtesy of Ampex Corporation, 111; Anderson-Jacobson, 53 (top left); Bank Americard, 118 (bottom); Burroughs Corporation, 50 (top), 51 (top left and top right), 102 (right), 135 (bottom), 145 (top left); California Computer Products, Inc., 128 (bottom); *Computers and People*, August 1975 © 1975 by Berkeley Enterprises, Inc., 32; Data General Corporation, 43 (left), 161 (top); Datapoint Corporation, 53 (center right), 55, 140; Digital Equipment Corporation, 21, 43 (right), 51 (bottom left), 54, 60, 106, 126, 137 (top), 138 (middle left, center left, and center right), 145 (bottom), 158; General Binding Corporation, 116; Carl Glassman, 114 (top); Hazeltine Corporation, 138 (bottom left); Hewlett-Packard, 50 (bottom), 122, 128 (top), 134 (top), 135 (top left), 136, 137 (bottom), 138 (top middle and bottom center), 150 (top left); Honeywell Inc., 108, 112, 125; Information Terminals, 97 (left bottom); International Business Machines Corporation, 46, 53 (bottom left), 91 (top), 102 (left), 131 (right), 134 (middle), 135 (top right), 138 (top left and bottom right), 141 (bottom), 145 (top right), 150 (top right), 161 (bottom); Kybe Corporation, 97 (top right, right bottom, and top left); Lear Siegler, Inc., 53 (top right), 134 (bottom); Maxell Corporation of America, 97 (top); Mohawk Data Sciences Corp., 95; National Cash Register Company, 4, 97 (top left), 142, 284; Nortronics Company, Inc., 18; Optical Scanning Corporation, 80; Pacific Northwest Bell, 132; Rotkin P.F.I., 19; Southern Pacific Transportation Company, 141 (top); Teletype Corporation, 51 (bottom right), 53 (bottom right), 131 (left), 138 (top right); Westinghouse Learning Corporation, 104.

# CONTENTS

# ELEMENTS OF COMPUTER CAREERS

# 1234567



# The computerized society

## THE MARVELOUS JOURNEY OF MILLIE FOGARTY

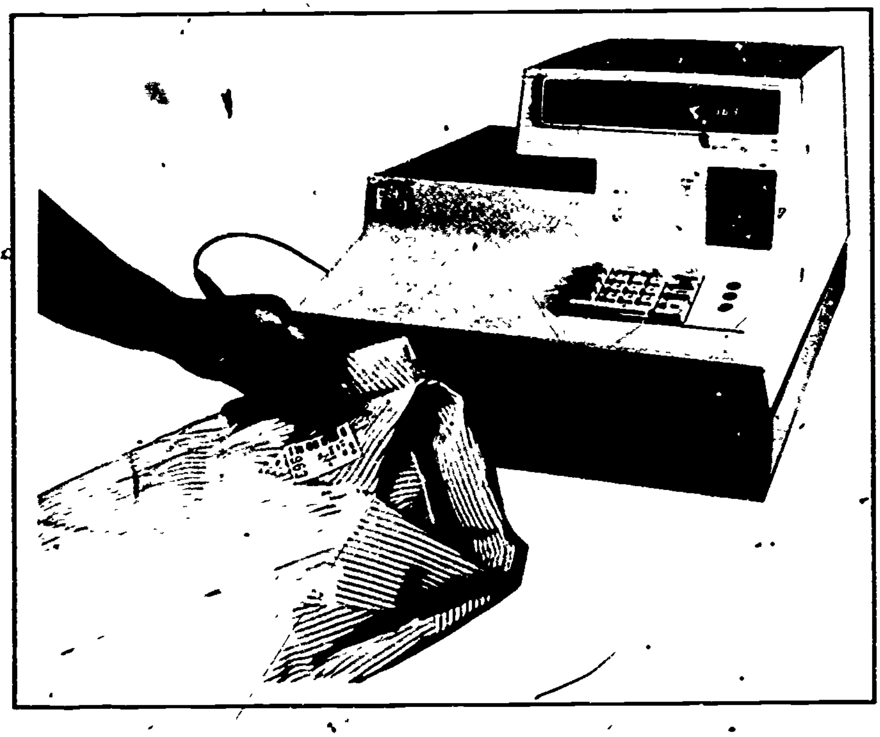
Millie Fogarty is a computer programmer. She works for a major computer manufacturer in Los Angeles. Frequently Millie is sent by her company to work on special programming projects in other cities. Millie is a "child of the 70's"—modern electronic computers were born before she was, and she takes them for granted. When she encounters a computer at the supermarket checkout counter, or if a computer-produced "voice" intercepts her call when she dials a wrong number, she scarcely notices. In fact, she depends on computers to make her life less complicated and more interesting. Let's follow Millie through a few days of work and leisure and note the impact of the computer on one life.

On Friday, Millie learns that she must fly to Oklahoma City to work on a special project. It is a program which simulates the drilling of an oil well. She is to be there by Monday morning, prepared to stay several weeks if necessary.

That means Millie needs to run some errands. Her old suitcase has traveled so much it is worn out. She'll need a new one. And she'll need to make plane and hotel reservations and withdraw some money from her checking account.

Saturday morning, her first stop is a department store. In the luggage department there are dozens of suitcases to choose from—flow-

**Fig. 1–1** A "reading wand" in use at a checkout stand



ered, striped, plain, large, small. Millie selects a medium-sized black suitcase and takes it to the clerk. As the clerk pulls off the magnetically coded ticket, he picks up a "reading wand"—a device that looks like a ball-point pen connected by a cord to the cash register. When he touches the wand to a magnetic strip on the ticket, codes for the color, price, stock number, and department number are read automatically and instantly into the electronic cash register. From there, the data goes to the store computer. At the end of the day all of that day's sales information is transmitted to the main computer at the national headquarters of the department store chain. This information is used to keep track of the inventory. When the store's supply of medium-sized black suitcases falls below a certain number, an order will be printed automatically for more to be sent from the warehouse. If the warehouse supply also gets low, the main computer will print a purchase order to be sent to the wholesale luggage supplier. This same sales information is used to help the store's buyers and managers make decisions, forecast future sales, and compute the monthly commission of the clerk credited with the sale of the suitcase.

Meanwhile, Millie hands the clerk her credit card, and again he brushes the wand across a magnetically coded strip located this time on the plastic card. In less than a second, the store's computer verifies her good credit rating and records the charge to her account.

There are several display lights on the screen of the electronic cash register. One of them, labeled "HOLD," has remained steadily dark. "What is the 'HOLD' light for?" Millie asks the clerk.

He grins, "If that light had come on I'd have called the police, because that means the credit card the computer is checking is a stolen one!"

As Millie leaves with her new suitcase she realizes the entire purchase took less than half the usual checkout time at other stores. Two waves of the magic wand—that's all there was to it.

Next, the airlines office. Several agents sit behind a counter, each operating a keyboard with a TV-like screen. Millie recognizes these keyboard-display units as "terminals," all connected to a central computer reservations system. She frequently uses a similar terminal in her work, although hers is connected to a different computer.

"May I help you?" inquires an agent.

"Yes," Millie replies. "I'd like a round trip ticket to Oklahoma City, leaving tomorrow night with open return." The agent checks a flight schedule.

Fig. 1–2 Computer terminals speed up airline reservations.

"Is 4:45 Sunday afternoon O.K.?"

"What time does it arrive?"

"That's flight number 172, arriving at Will Rogers International Airport at 9:10 P.M."

"Fine. Is there a coach seat available?"

The agent keys into the terminal the flight number, destination, and date desired. In a second or two the status of that flight flashes on the screen: five seats left in the coach section and nine in first class.

"Your name, please?" the agent asks, looking up from her screen.

"Millie Fogarty."

She keys in the name and class desired, and immediately the screen flashes "CONFIRMED." As the agent starts to make out the ticket, the control computer automatically reduces the number of coach seats available on that flight to four, and adds Millie's name to the passenger list.

Millie's bank is just down the street from the airlines office, so she stops in for some money. She writes a check for $150 cash and hands it to the teller. He picks up a telephone, dials the bank's computer, and reads the checking account number, speaking slowly and distinctly. In a few seconds the computer responds in a mechanical voice: "The balance in that account is $340.14." The check is cashed. Hmmm, doesn't leave much money in the bank to cover the bills routinely paid by her bank's computer each month. However, Millie knows her paycheck soon will be deposited automatically in her account, so that new white sports car is secure for another month!

Lighthearted, Millie stops on her way home at a Howard Johnson motel and asks for a reservation at the downtown Howard Johnson in Oklahoma City. The desk clerk goes to a terminal which resembles a large electric typewriter and types Millie's name, date of arrival, type of room requested, and the identification number of the Oklahoma City motel..The terminal automatically types a confirmation for the room, which the clerk tears off and hands to Millie.

Sunday morning both Millie and the airline's computer are busy preparing for the flight. The computer has stored information on all available airplanes, with their maintenance records, schedules, and capacity; it also maintains a file on every pilot and each cabin attendant, with medical status, aircraft qualifications, hours flown each month, and personal and payroll information. So, while Millie packs, the computer uses all this information to juggle the flights to be made with the airplanes and crews available, and selects the most efficient and economical combination possible for Millie's flight. Because of the computer's homework, Millie will not find herself flying in a jumbo jet from Los Angeles to Oklahoma City with only ten other passengers!

When Millie arrives at the airport she places her bags on a conveyor belt at the ticket counter. An attendant puts a tag on each bag and gives Millie the stubs. At the same time, speaking into his headset microphone, he tells the computer the flight number and destination for the bags. The computer interprets the words he speaks and routes the bags to the proper airplane on a network ot conveyor belts.

In the control tower the computer, still working away on Millie's flight, gives the air traffic controllers a constant picture of the air traffic in the area. Her plane is cleared for takeoff, and leaves safely and on time. The landing in Oklahoma City is similarly smooth, even though a thick fog obscures the airport. The plane is landed "blind," guided only by instruments and the cockpit computer.

Millie loads her bags into a waiting taxicab and heads for her downtown motel. On the way, she notices that traffic appears to be moving very smoothly, with few stops at traffic lights. These lights, she senses, must be controlled by a computerized traffic control system. She is correct. The traffic lights at 33 major intersections are linked to a computer in the city's municipal building. Electronic vehicle detectors in the pavement at intersections collect data on the number and speed of cars and send them to the computer. The computer constantly analyzes the data and selects the best signal pattern to improve traffic flow and reduce the number of stops for all 33 intersections.

By the time Millie checks into her room, she is tired. There is a good play in town, but she decides to wait and get a ticket for the following evening instead. After she unpacks, she sees a card on the television set announcing "in-room movies" on channel 5, for $2.00. She calls the desk and asks the clerk to start the movie—a new one she'd been planning to see—and relaxes on her queen-size bed to watch. A computer handles the transmission of the movie to the TV set. The clerk simply keys in the appropriate room number and adds the $2.00 charge to the room bill.

The next day, during her coffee break, Millie goes to a Ticketron agent in a downtown pharmacy to reserve a ticket for that night's play. The operator keys in the date and the play. The terminal responds that the play is sold out for both Monday and Tuesday nights' performances, but Wednesday is O.K. She decides to get a ticket for Wednesday. The terminal confirms by printing the ticket, which Millie charges to her BankAmericard.

In her own state, Millie has used Ticketron to reserve space in a campground, get tickets to football games and concerts, and even make reservations for a wilderness canoe trip.

## THE COMPUTER CAN . . . AND SOMETIMES CANNOT . . .

Millie Fogarty, and you and I are surrounded by computers, but how often are we even aware of their influence? Sometimes we perceive that transactions are quietly being speeded up or new services becoming available, and sense that computers are there serving us, slavelike. Sometimes there is a "foulup," and a poorly programmed computer causes us moments of frustration or irritation. But, whether realized or not, computers have become an indispensable part of our daily lives, used and depended upon in much the same way as we depend on our telephones and automobiles.

None of the ways computers were used during Millie's marvelous journey are fictitious. Not only are computers being used in all of these ways, but in ways even more marvelous.

The new age of the computer has been called by some the second Industrial Revolution; by others, the Computer Revolution or the Cybernetic Revolution. The first Industrial Revolution, in eighteenth

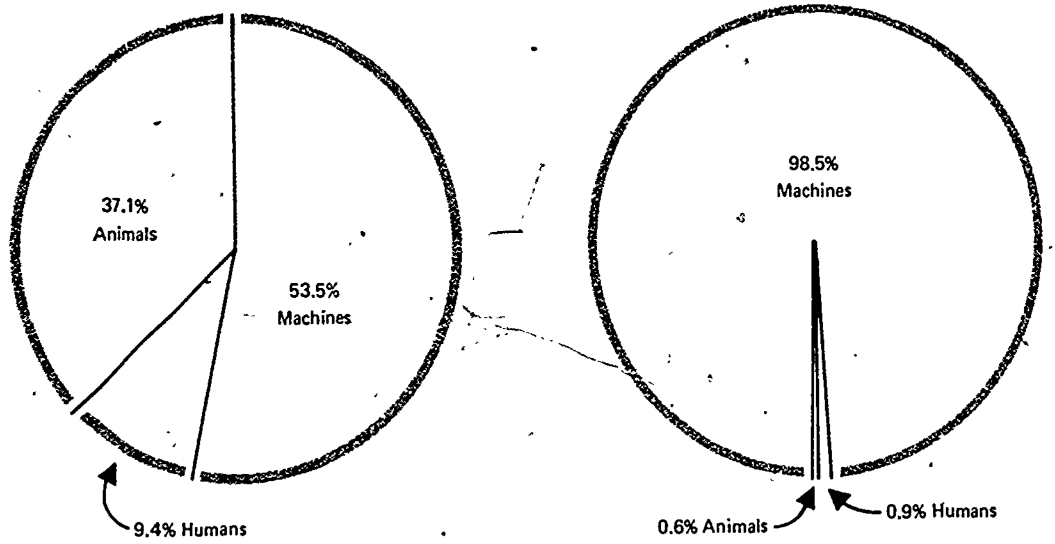


Fig. 1-3 Change in division of total horsepower hours as a result of first Industrial Revolution*

century England, had a tremendous impact on society. Machines released humans as beasts of burden. They amplified and extended the power of human (and animal) muscles and led to a dramatic upheaval in social and economic structure. The implications of Fig. 1-3 perhaps express this clearer than words can tell.

We are living through a similar upheaval due to the advent of the computer.† Computers do for the *brains* of men and women what machinery did for their muscles—they amplify and extend human power. Computations that people would find tedious, time-consuming, and repetitious are carried out quickly and effortlessly by the machine, with no errors.

Although the impact of this upheaval has been recent, the computer really is not a unique development of the twentieth century. A long history of developments, beginning 3,000 years ago with the abacus, is behind this development. Early in the seventeenth century, a device for multiplication was developed, and shortly afterward the

* Data from W. H. G. Armytage, *The Rise of the Technocrats: A Social History* (Toronto: University of Toronto Press, 1965).

† For some thought-provoking discussions of this upheaval, read *The Computerized Society* by James Martin and Adrian R.D. Norman, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1970; or *Computers and Society* by Stanley Rothman and Charles Norman, Science Research Associates (a subsidiary of IBM), Chicago, Ill., 1970.

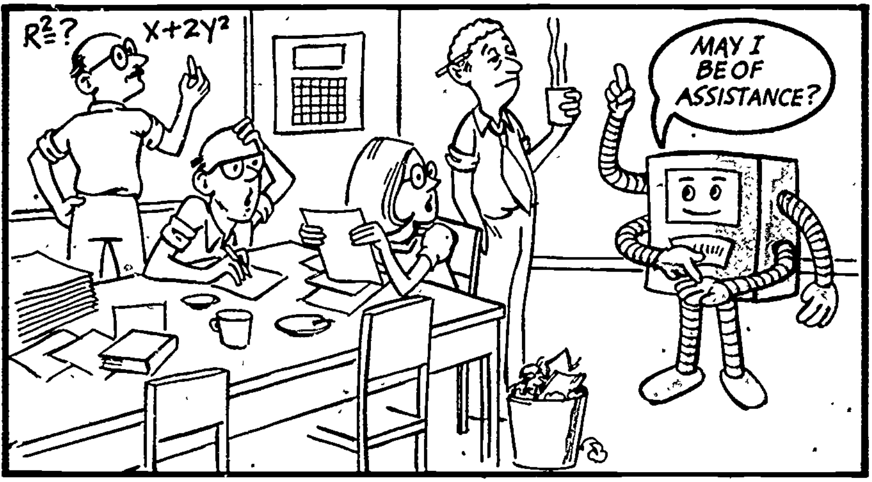


**Fig. 1–4** People may work for years to solve a problem which a computer can solve in seconds.

slide rule. The first true calculating machine was invented in 1642 by Blaise Pascal. The principles used in Pascal's computer are still used in computers today. A machine to mechanize trigonometric and astronomical calculations appeared in 1694. The idea of using punched cards to supply information for controlling machines also was developed in the seventeenth century. Charles Babbage, a British mathematician whose name is usually linked with the development of the first modern computer, relied upon these historical concepts and developments in building his prototype computing machines. From his work on these, he then developed the idea for the first modern computer; but his work lay forgotten because the necessary technology to build the required gears and other mechanical parts did not exist. Therefore, it was not until a century later that work continued. Then electric circuits were used instead of mechanical devices.

The modern computer, based on electronic rather than mechanical or electro-mechanical principles, *is* a product of the twentieth century. But all the components of this modern development already existed: the memory unit, for information storage; the arithmetic unit, for mathematical calculations; and the control unit, for translating information into a form the computer can read.

Many good and certainly more detailed histories of computer technology exist. A suggested one is *A Computer Perspective*, by Charles Eames, published in 1973 by the Harvard University Press.

Let's consider why people wanted to develop computers in the first place. What are they uniquely capable of doing that can help humans? The computer can

- do arithmetic in billionths of a second;
- solve complicated problems with never an error;
- work without break at a task for hundreds and hundreds of hours, no matter how boring or repetitive or difficult it might be;
- store vast quantities of information and retrieve it instantaneously;
- solve (theoretically) any problem for which all necessary decisions and variables can be defined and stated in quantitative (numerical) terms.

This last *capability* brings us also to consider what a computer's *limitations* are. When we specify decisions of a definable and quantifiable nature, what do we mean? The decision of whether or not a person is eligible to try out for the position of violinist in your city's symphony orchestra is one which a computer is capable of making. All variables are quantifiable. A person is eligible if he or she is over 18, has taken violin lessons for at least four years, has previously played in an orchestra for at least one year, and is willing to devote ten hours per week to orchestra rehearsals. If the numbers all fit, the person passes.

However, *selection* for the orchestra from those eligible cannot be done by a computer, for this is a decision that must be made by a musician with a highly trained ear. The factors (or variables) bearing on the decision in this case are purity and quality of tone, finger dexterity, rhythm, timing, ability to read and interpret music, and the quality of performance of all others auditioning at the same time. Obviously, many of these variables cannot be quantified—that is, automatically stated in terms of a number. Rating depends, instead, on the personal judgment of the listener. If seven different listeners judged the performer, seven different rating scores might be assigned.

What kinds of things *cannot* be quantified, then? Perhaps medical diagnosis? We are accustomed to the notion that it takes the practiced eye and trained intuition of a real live doctor to make an accurate diagnosis. Yet, computers have been programmed to interview patients, analyze the results of lab tests, and finally make a diagnosis and recommend treatment. The computer diagnoses have been as accurate or more so than those made by doctors seeing the same patients!

Perhaps a decision like "Who gets welfare payments?" should be made by humans. Yet, all requirements for eligibility can be quantified easily. Even decorating an office building can be quantified; decisions on color, style, and arrangement of furniture can be made by a computer, based on the number and type of occupants of the building, the type of businesses housed in the building, the climate, number and placement of doors and windows, and so on. Whether or not a person *likes* the decor, however, is purely a matter of individual taste, just as it would be if an interior decorator planned the whole thing.

It seems, then, that decisions that cannot be acceptably quantified are those that have to do with the aesthetic effect of a work of art (a painting or a symphony, for example), individual taste, or personal values. Can you think of others?

The power of the computer to relieve people of tiresome or complicated mental work has led some to make comparisons between the computer and the human brain. The computer can do more arithmetic in one second than a person can do working 40 hours a week for 11 years. But the human brain can store 100,000,000,000 times more information than the computer. Such comparisons are unfortunate, because they lead many to the conclusion that the main differences between peoples' brains and computers are in speed or memory capacity. They ignore the human ability, for example, to interpret the meaning of spoken words through facial expressions, gestures, past experience, and emphasis, or the human abilities to handle unforeseen emergencies, produce new ideas, establish values, select goals, establish and maintain emotional relationships, and make sound judgments based partly on "intuition."

Consider the five-year-old girl who unexpectedly found herself a witness to a kidnapping in the playground of a neighborhood park. She knew "instinctively" that the two men who forced a protesting child into a car were "bad" men. How? She sensed that it was inappropriate for men dressed like the kidnappers to be in a childrens' playground. She remembered the warnings of her parents about strangers. She took in information from many sources at once (saw the frightened look on the child's face, heard the gruff commands of the kidnappers, witnessed their rough handling of the child, heard and saw the hasty getaway, observed the color and type of car). The girl processed all of this information instantly, and she was able to communicate these facts and perceptions to police officers when she ran for help.

While this may not seem to be an amazing feat for a five-year-old child, getting a computer to perform similarly would boggle the mind of a computer programmer! To accomplish the same task, a computer would have to be capable of receiving diverse information by seeing people and actions and hearing sounds and voices. It would have to simultaneously retrieve pieces of information from memory, interpret facial expressions, gestures, and tone of voice, and sense "what is appropriate" based on past experience—all in the space of a few seconds. Then it would have to judge what action to take (get help!) and be able to report all that happened on demand. While all of this may someday be technically possible for computer, it would require a programming effort of such magnitude and complexity, and machinery of such refinement and sophistication, as to make the task overwhelmingly forbidding—not to mention expensive!

Obviously, we are not going to waste much effort endeavoring to create the mechanical equivalent of a five-year-old child, let alone an adult human being. The kinds of things we can and do use computers for are considerably simpler and more practical.

What kinds of things then can be called limitations of the computer? Some rather fundamental ones are these:

- For even some relatively simple problems, setting up the computer to solve them requires high costs and long delays. For example, preparing a computer to do what the five-year-old child did in recognizing a crime and observing and reporting relevant details would require a tremendous effort and cost. The results would probably be inferior to the child's performance also.
- Even though computer costs have steadily decreased, many potential applications are too costly for computerization.
- As efficient as a computer is, it cannot be "creative" and itself discover new ways to solve problems more efficiently. It can only solve problems according to the carefully structured way it has been programmed.
- A computer can work only with exact quantities. It is helpless in dealing with "maybe," "a little bit," "usually," "I'm not sure yet," or "tall, dark, and handsome." A computer must be told exactly how many inches is "tall," and the length of nose, texture of hair, weight, thickness of eyebrows, and other characteristics which make a man "handsome."

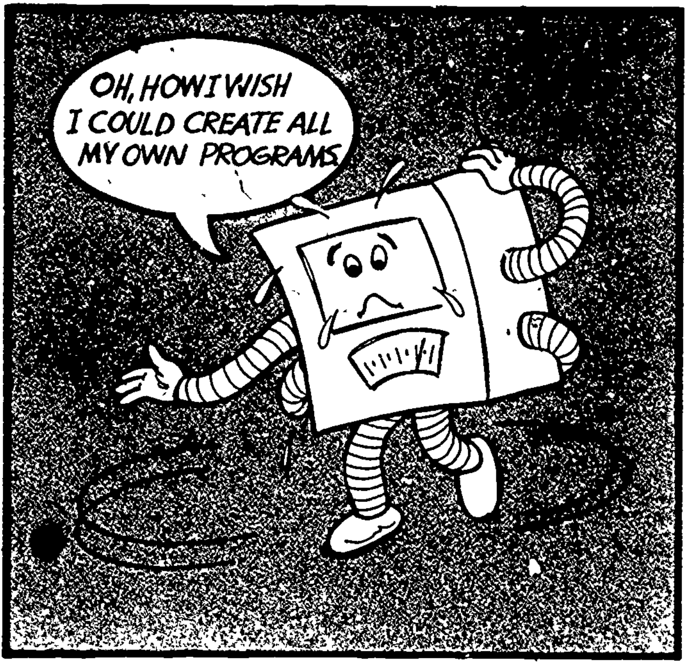


Fig. 1–5 People can think, imagine, and create new things—computers can only process data according to instructions.

It is important to consider that, just as advances in technology have overcome some former limitations, so might these limitations be minimized in the future.

Projected or hypothesized uses for the computers of the future will also require human beings to ponder some very important issues. These include protection of individual privacy, computerized control of human behavior,* and how best to utilize the increased leisure made possible by computer automation. We are faced with the necessity of controlling the uses of the computer so as to improve the quality of life rather than to increase its problems.

## THE COMPUTER IN THE WORLD OF WORK

In spite of the limitations of computers, there are virtually no areas of activity in the world today that have not been influenced by the com-

* Such a technologically possible situation is the subject of Michael Crichton's fascinating—and frightening—book, *Terminal Man,* Bantam Books, Inc., New York, 1973.

puter to some extent—often to a very great extent. You cannot get through a normal day without being touched by this influence many times, whether or not you are aware of it. If you receive mail, order books, attend classes, use the telephone, take medicine, eat food purchased in supermarkets, have a driver's license, watch television or listen to the radio, shop in department stores, travel by train or airplane, use a credit card, write checks, apply for a job, have a savings account, read the newspaper, use the library, or apply cosmetics, you already depend on computers, and information files are being built about you now, based upon that dependency.

Because of the influence of the computer, there have been predictions that as high as 50 percent of all jobs available ten years from now will not have even existed today. Whether or not this is an accurate prediction, we can be sure that most jobs will be influenced by computers.

Let's take a closer look at some of the areas in which computers are helping men and women to do their jobs better, or freeing them for more enjoyable activities.

## Science

It is in the areas of scientific research, engineering, and space science that computers have had the earliest, greatest, and most constant impact. No scientist worth his or her test tubes would consider doing important scientific research without a computer—as basic and essential a tool, now, as a microscope. The computer has made possible experiments of greater complexity than ever before. In some cases, it even eliminates the necessity of doing experiments, as the computer can "simulate" complex scientific procedures. Some scientific research simply cannot be performed at all without such a simulation.

For example, the energy crisis has prompted a serious search for ways of generating thermonuclear power. Astrophysicists, therefore, have increased their studies of nuclear reactions and other events that take place inside stars. Obviously they cannot make stars in their laboratories in order to study them, but they can and do use the computer to test possible models of star interiors. The experimenter can provide hypothetical information to the computer about the materials in each part of the star, the reactions taking place, and the flows of heat and radiant energy between each part of the star. The computer calculates and reports the results of the reactions occurring in each part of the star, the effect on every other part, and finally the effect of these reactions on the outer boundaries of the simulated star.

Every product as well as each part in a product must be designed before it can be manufactured. The computer has become an amazing tool for engineers and designers in this capacity. Chemical engineers designing plastics, for example, can analyze on the computer the combinations of chemicals they think are needed before they ever begin to produce the plastic material. Electrical and mechanical engineers can also use the computer to produce a model of a process and check its accuracy and reliability before it is used. Designs which used to take engineers hours to draw are now done in seconds by a computer.

The future generation of computer experts will work with engineers to design computer equipment to improve designing and drawing methods. One computer tool now available, called "SKETCHPAD," can already turn an engineer's rough draft drawing into an accurate design drawing right on a computer display screen.

The Lunar Expedition Module (LEM) which took Apollo astronauts to the surface of the moon was controlled by an extremely sophisticated computer system. It accumulated, analyzed, and processed data to keep the craft on a predetermined flight plan at all times. In space flight, navigation is more difficult than simply flying an aircraft between two fixed points on earth. This is because the destination (moon) and the point of origin (orbiting spacecraft) are both moving rapidly. The computer must constantly recalculate trajectory and adjust the controls to keep the module on course.

A recent NASA plan is to develop a robot that will be able to roam Mars with little guidance from Earth, avoiding obstacles, selecting subjects for photographs, finding interesting geological samples, and performing chemical analyses on them. Since it takes twenty minutes for a radio signal to travel from Earth to Mars and back again, a robot with its own self-contained computer could gather and process more data with greater efficiency than could one that depended on receiving each of its commands from Earth.

When Mariner 9 reached Mars in 1973, it sent 7,200 photographs back to Earth, so that scientists could study changes on the surface of the planet. Analyzing the photographs was an overwhelming task made manageable by the computer, which stored, classified, organized, and compared each photograph. Scientists were then able to study photographs taken of the same region at different times to discover changes that had taken place in the planet's surface.

## Business and finance

At first, the business world used computers as super-clerks. Only routine record keeping, and tasks such as calculating salary checks or sending bills to customers, were assigned to the computer.

More recently, the computer serves as the heart of an "information system," in which all aspects of a business are interrelated and cross-indexed. This provides immediate information to help managers plan and make decisions.

The capacity of the computer to accurately store and retrieve masses of information makes it possible for businesses to manage the numerous and increasingly complex types of information generated daily in business operations.

The computerized cashless, checkless society may soon be a reality if the plan of the Federal Reserve Board materializes. Anticipating an impossible overload of 48 billion checks per year by 1980 (double the number of checks written in 1971), the Board proposes an electronic banking system. It would be comprised of a central bank with 44 huge computers around the country to handle the load. If banks cooperate, the number of checks written could shrink to a mere 22 billion by 1980. As a step in this direction, paychecks could be eliminated as local employers began to transfer money electronically directly to the employees' checking accounts. These employees, customers of the bank, would preauthorize automatic payments (without checks) for such items as insurance premiums, utility bills, and new white sports cars. Thus, the normal monthly bill-paying would be handled automatically and electronically. Such a bill-paying system is already in operation in most of California's banks, and other banks around the country are gradually joining. The Federal Reserve Board's 44 computers, once installed, would tie all commercial banks together in a centralized network.

The next step in eliminating checks is also already in operation in many locations: the installation of electronic "point-of-sale" terminals, or electronic cash registers, in retail stores such as supermarkets, department stores, and even small businesses. When Millie purchased her medium-sized black suitcase, she encountered such a terminal. Activated by a credit card, the terminal automatically charges the customer's account and can automatically credit the merchant's account at the bank.

In 1973 the grocery industry made progress on this last step by implementing a standard Universal Product Code (UPC) for marking supermarket products. Soon, markets will be able to link all their checkstands to a computer. The checker pulls each item across a scanner imbedded in a slot in the checkstand, then drops the item in a bag. The scanner reads the UPC code on the label and transfers all information to the computer. Once again, the customer's account can automatically be charged and the grocer's account credited for the total amount of the sale. Such a system can speed up checkout time by half. By the end of 1975, most grocery items will be marked with the UPC, and many markets will be turning to electronic computer-linked point-of-sale terminals.

A system even more convenient was launched in 1973 in Louisville, Kentucky. Called Call-a-Mart, families may join for a $5 fee. Shoppers place orders by telephone, using code numbers from a catalog. A Call-a-Mart operator keypunches the order into cards, which are then read into a computer. Workers along computer-guided conveyor belts fill the order. The computer also picks the best delivery route from store to shopper, and registers the most convenient time for the shopper to receive and pay for the groceries.

Antiquating the traditional cash register approach in another way is a system which can actually recognize the spoken names and prices of grocery items, display them for the customer, and then print a receipt. Threshold Technology Inc. calls this recent development the Voice Activated Checkout System.

A new kind of executive using the computer is the farmer—no longer a folk hero, but an agribusinessman. One 15,000-acre farm in

Fig. 1-6 Clerks use reading wands at point-of-sale terminals for faster and more accurate checkout service.

Fig. 1–7 From this boring assembly line to streamlined, automated process

California and Arizona is hooked by terminal to a computer in Houston, Texas. This computer produces monthly reports analyzing, by area and by crop, such variables as fertilizer, cultivation, tree replacement, insect control, equipment cost, and harvesting. Planning is guided by twenty-year projections also supplied by this computer. Centralized electronic consoles direct the watering system for the farm's 26 different crops.

In Colorado, a 35,000-animal cattle feed lot uses a computer to automatically calculate, blend, and dispense the proper proportions of food to each feeding pen. In Pennsylvania, a computer has helped to double the milk output of a quarter million cows through computer-guided selective breeding. In Iowa, a crop-planning computer tells farmers how many acres to plant of corn or soybeans in the spring, and helps them plan for farm enlargement or reorganization.

### Industry

The potential for "cybernation"—total automation and control of formerly manual tasks—is greater in the field of industry than in any other area. Industrial processes that transform raw material to finished product use machines that require constant monitoring and control. This can be efficiently and economically assigned to a computer.

Automated control of machine tools and of the process equipment used in the chemical and petroleum industries is perhaps the biggest application of computers in industry. A large computer may monitor the consistency and potency of a batch of chemicals to catch discrepancies. It may control the operation of a machine tool by precisely positioning the tool for cutting or drilling, and then handling the cutting.

A useful team of human and machine selects parts from the 46,000 items in the Caterpillar Tractor Company's huge Denver warehouse. When orders for parts come in, a vehicle with a person inside is automatically guided by a computer down the correct aisle and raised to the correct bin, where the operator hand-picks items (since people can still pick and grasp different shapes better than can machines). The system allows employees to accomplish more work and get more rest time.

Entire manufacturing operations could eventually be taken over by automated controls. Raw metal rolled in at one end of a football-field-sized plant would be treated, formed, cut, packaged, and then rolled out the other end as safety pins ready for sale without the touch of human hand.

In spite of this potential for automation in both accounting and production, over half the labor force is still employed in industries that are *not* automated. The concern that computers would put people out of work apparently is unfounded—while some workers are displaced, new jobs are created and some of the old jobs change. In the fully automated industries (such as petrochemicals) assembly-line drudgery no longer exists. In contrast, workers must now exercise more control and responsibility than they did on any assembly line.

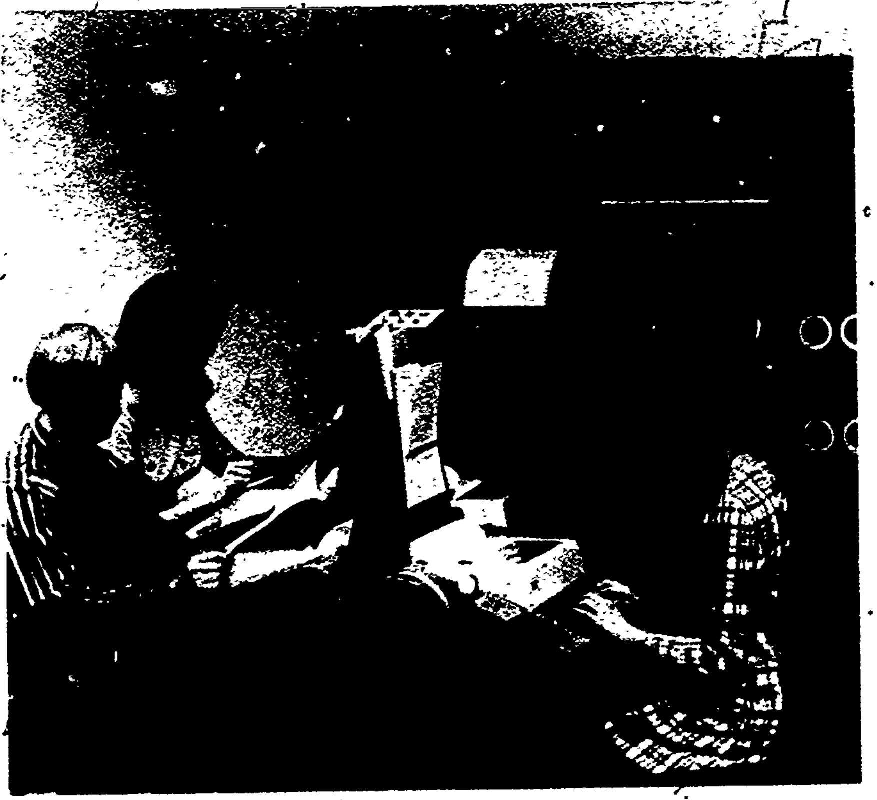
## Education

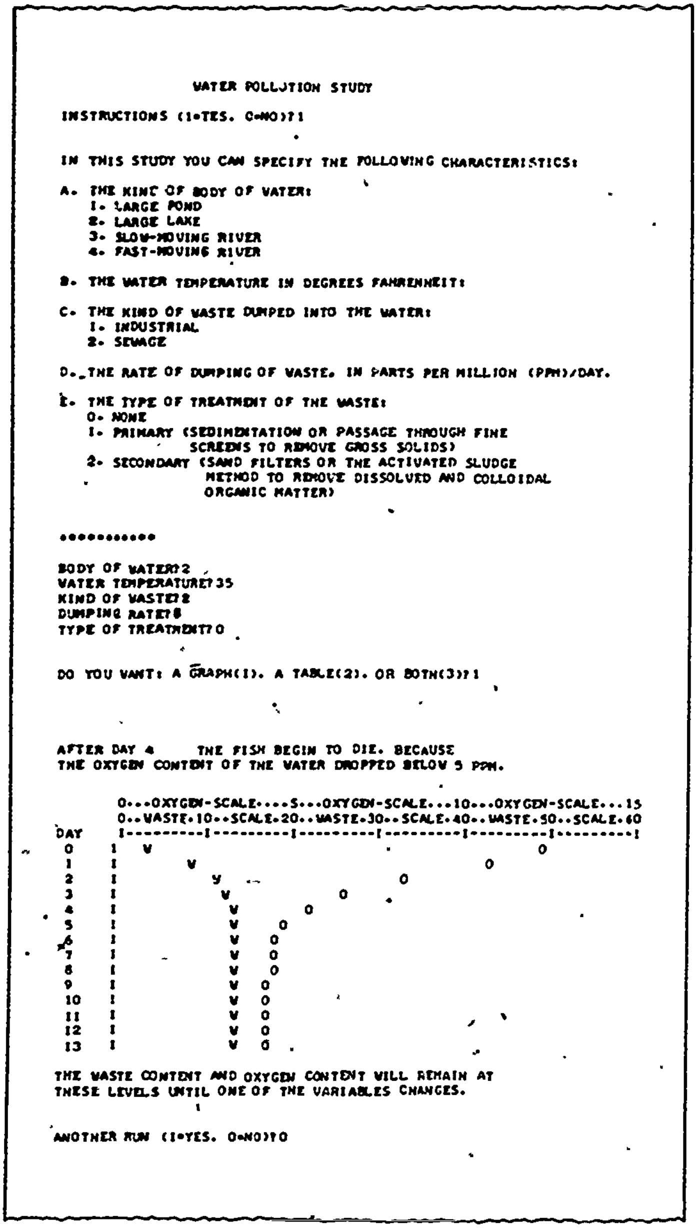
Today's school administrator finds ways to use the computer in scheduling classes, forecasting school budgets, and even planning schoolbus routes. Like any executive, she or he has long since computerized routine accounting, payroll, and record keeping jobs. University administrators find the computer to be even more indispensable, dealing as they do with the same tasks on a larger and more complex scale, and making long-range plans in a time of tight budgets and decreasing enrollments.

It is in the instructional area, however, that the most exciting things are happening. In a classroom where the computer is a fully accepted and utilized instructional tool, you may observe the following scene.

Several students are gathered around a clattering typewriter terminal, arguing over how they should respond to the computer's questions. The program they are running simulates the dumping of organic wastes into their water supply. They are studying the effects of such pollutants on various bodies of water. The students decide what type of water, waste, and treatment they want to try out, and the computer tells them the results. The printout on the following page shows the computer's response to a particular set of data. (Student input and responses in this and other similar examples are underlined.)

**Fig. 1–8** Students find classroom work expanded by the use of computers



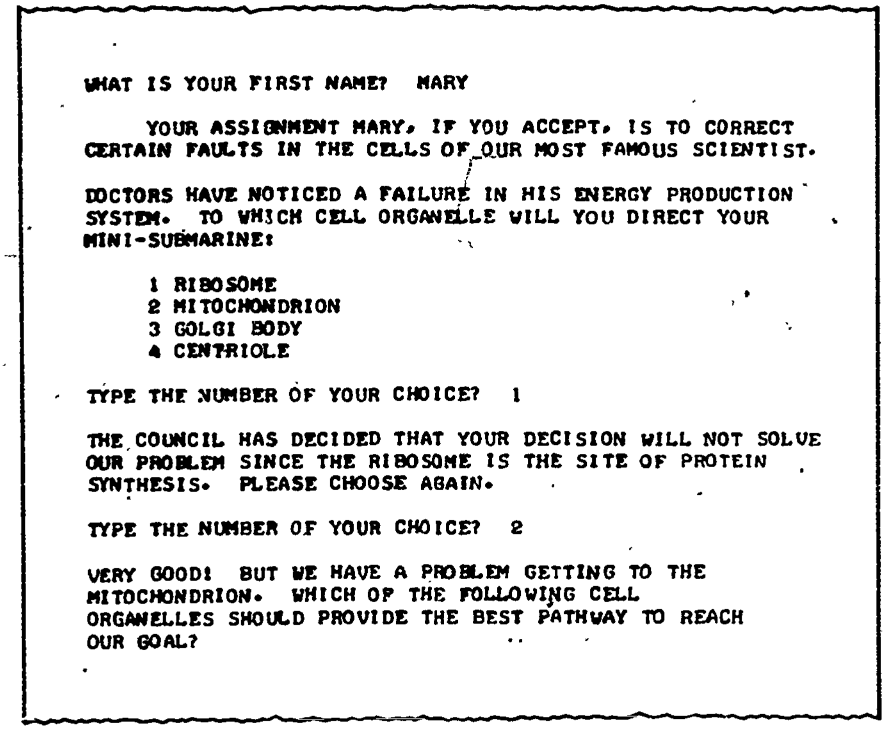


```
                WATER POLLUTION STUDY

INSTRUCTIONS (1=YES. 0=NO)?1

IN THIS STUDY YOU CAN SPECIFY THE FOLLOWING CHARACTERISTICS:

A. THE KIND OF BODY OF WATER:
   1. LARGE POND
   2. LARGE LAKE
   3. SLOW-MOVING RIVER
   4. FAST-MOVING RIVER

B. THE WATER TEMPERATURE IN DEGREES FAHRENHEIT:

C. THE KIND OF WASTE DUMPED INTO THE WATER:
   1. INDUSTRIAL
   2. SEWAGE

D. THE RATE OF DUMPING OF WASTE. IN PARTS PER MILLION (PPM)/DAY.

E. THE TYPE OF TREATMENT OF THE WASTE:
   0. NONE
   1. PRIMARY (SEDIMENTATION OR PASSAGE THROUGH FINE
              SCREENS TO REMOVE GROSS SOLIDS)
   2. SECONDARY (SAND FILTERS OR THE ACTIVATED SLUDGE
                METHOD TO REMOVE DISSOLVED AND COLLOIDAL
                ORGANIC MATTER)

***********

BODY OF WATER?2
WATER TEMPERATURE?35
KIND OF WASTE?2
DUMPING RATE?8
TYPE OF TREATMENT?0

DO YOU WANT: A GRAPH(1). A TABLE(2). OR BOTH(3)?1



AFTER DAY 4      THE FISH BEGIN TO DIE. BECAUSE
THE OXYGEN CONTENT OF THE WATER DROPPED BELOW 5 PPM.

        0...OXYGEN-SCALE....5...OXYGEN-SCALE...10...OXYGEN-SCALE...15
        0..WASTE.10..SCALE.20..WASTE.30..SCALE.40..WASTE.50..SCALE.60
DAY     I---------I---------I---------I---------I---------I---------I
 0      I  W                                             O
 1      I       W                                 O
 2      I          W                    O
 3      I           W             O
 4      I            W        O
 5      I            W     O
 6      I            W    O
 7      I            W    O
 8      I            W    O
 9      I            W   O
10      I            W   O
11      I            W   O
12      I            W   O
13      I            W   O

THE WASTE CONTENT AND OXYGEN CONTENT WILL REMAIN AT
THESE LEVELS UNTIL ONE OF THE VARIABLES CHANGES.

ANOTHER RUN (1=YES. 0=NO)?0
```

**Fig. 1-9** Sample output from science simulation POLUT*

At another terminal a student is busy learning some scientific terms, and actually enjoying it! Here is the conversation she is having with the computer:

* Program POLUT is available in the "Huntington II Simulation Package—POLUT" from Digital Equipment Corporation (146 Main Street, Maynard, Massachusetts).



```
WHAT IS YOUR FIRST NAME?  MARY

     YOUR ASSIGNMENT MARY, IF YOU ACCEPT, IS TO CORRECT
CERTAIN FAULTS IN THE CELLS OF OUR MOST FAMOUS SCIENTIST.

DOCTORS HAVE NOTICED A FAILURE IN HIS ENERGY PRODUCTION
SYSTEM.  TO WHICH CELL ORGANELLE WILL YOU DIRECT YOUR
MINI-SUBMARINE:

     1 RIBOSOME
     2 MITOCHONDRION
     3 GOLGI BODY
     4 CENTRIOLE

TYPE THE NUMBER OF YOUR CHOICE?  1

THE COUNCIL HAS DECIDED THAT YOUR DECISION WILL NOT SOLVE
OUR PROBLEM SINCE THE RIBOSOME IS THE SITE OF PROTEIN
SYNTHESIS.  PLEASE CHOOSE AGAIN.

TYPE THE NUMBER OF YOUR CHOICE?  2

VERY GOOD!  BUT WE HAVE A PROBLEM GETTING TO THE
MITOCHONDRION.  WHICH OF THE FOLLOWING CELL
ORGANELLES SHOULD PROVIDE THE BEST PATHWAY TO REACH
OUR GOAL?
```

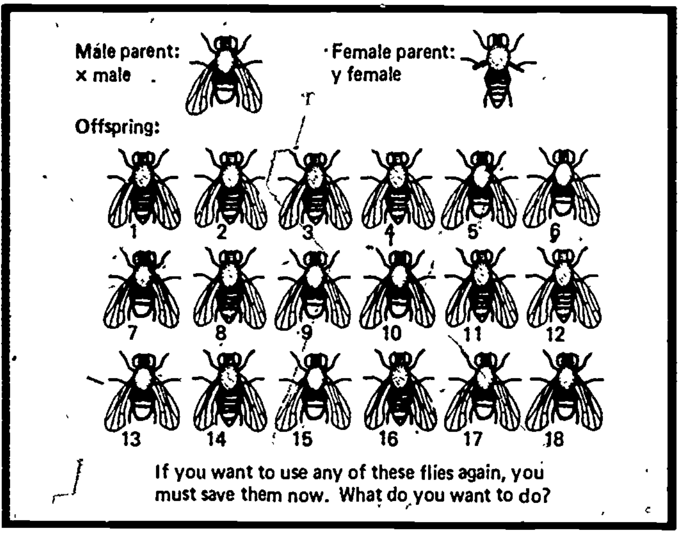
**Fig. 1–10** A typical interaction with a computer instructional program*

Two other students at a computer terminal are examining a screen which displays pictures of the offspring of two fruit flies. They are simulating experiments in breeding successive generations of fruit flies. The students select two of the fruit fly offspring pictured to breed again. Then the terminal displays the resulting new generation of fruit flies.

Other students are taking turns using computer programs they have written to quickly calculate the complex data they have gathered during an experiment in their biology laboratory.

At the same time, terminals are in use in mathematics, social studies, language arts, business education, and even in art and P.E. In some of these classrooms, the computer is directly involved in com-

* James Friedland, "Computers in Secondary Science," *Computers in the Curriculum: A Book of Readings* (Portland, Oregon: Northwest Regional Educational Laboratory, 1974), p. 107.



Male parent: x male

Female parent: y female

Offspring:

If you want to use any of these flies again, you must save them now. What do you want to do?

**Fig. 1–11** Sample print-out of a computer fly-breeding simulation*

puter-assisted instruction, as in the examples given. In others, the computer "manages" instruction. That is, students are tested and scored by the computer, which then writes a learning prescription for each student, especially geared to that student's own abilities, needs, and previous learning. The prescription tells the teacher and student what lessons the student should study next. It may assign a television program rather than a book if the student learns more easily that way. After each lesson the student takes another test, and the computer reevaluates the student's progress and writes another prescription.

Such uses of computers can add valuable time to teachers' days, freeing them for more individual tutoring or counseling with students. When relieved of the duller aspects of teaching, teachers can also spend more time individualizing a program of instruction for each student. With the help of the computer, a teacher could have each student proceeding at a different rate. The teacher would have more time to work with individuals, wherever each student might be in the course.

In universities, computers are also used for computer-assisted instruction and computer-managed instruction. Here research is also a major use. In most universities, computing is concentrated in the three areas of administration, instruction, and research. Heavy emphasis is on research, where the computer is an invaluable tool. Much of the important research done in this country is conducted in univer-

[*] "Computer-Assisted Instruction: Two Major Demonstrations," *Science*, Vol. 176, 9 June 1972.

sities, to such an extent that the university computing centers have a large body of "standard" computer programs to process the data and statistics generated in research activities.

- At least two dozen major areas of study are using computers at the university level—from architecture to forestry to religion. Universities used to include Latin as a mandatory entrance requirement. Now several universities require proficiency in at least one computer programming language—surely a more practical requirement in the 70's!

## Medicine

For about $50 you can now receive a complete physical as thorough as the best $150 checkup, in about two hours and without seeing a doctor. All over the country, Automated Multiphasic Health Testing (AMHT) centers offer a highly complex, totally automated medical checkup.

At the first stop, the computer (through questions flashed on a display screen) quizzes you to establish a "medical history." The computer zeroes in on "suspicious" responses and asks for more information. For example, the question "Do you have pain?" might appear. Simply by touching the screen at the appropriate point, you can answer yes or no. If your answer is "yes," a nude body appears on the screen and you are asked to touch the spot where you feel pain.

The system also records the results of many fully automated medical tests. It includes those for blood pressure, lung capacity, hearing, and electrocardiography. It also processes X-rays and urinalyses. The computer then prepares a detailed report for your doctor, and flags any findings that seem abnormal. The doctor then schedules an appointment with you to go over the report.

In hospitals' intensive care wards, nurses can keep close watch over many critically ill patients at the same time through a master console at the nurses' station. A computer monitors all the "vital signs" of each patient and instantly alerts nurscs and doctors to any change in heart rate, blood pressure, breathing, and so on.

Meanwhile, in surgery, a serious operation is underway. The surgeon pauses frequently to check a display screen which provides up-to-the-second information on the patient's condition. While the operation is being performed, a huge number of values are collected and analyzed, and a built-in signal warns the surgeon of critical situations.

One of the most amazing modern developments in medicine is the transplantation of organs in the human body. You are probably most aware of heart transplant operations in which a person with a fatal heart condition has the damaged heart replaced with a strong one. What you may not know is that much of the research leading to heart transplants was processed by a computer.

Did you know that a computer can read heartbeats and can report heart defects? In one scientific experiment, "phonocardiograms" were taken of several hundred children in order to gain knowledge about irregular heartbeats. It was a computer that organized and analyzed the data. A computer can listen to hours and hours of human heartbeat more efficiently (and certainly more patiently) than a doctor, and greater numbers of individual heartbeats can be analyzed by the computer. Scientists and doctors hope that this research will allow a doctor to diagnose a heart problem immediately. Thus lifesaving treatment can begin more quickly.

A computer in Bristol, England, has taken over the task of matching donated kidneys for patients awaiting a transplant. There is a severe shortage of donor kidneys, and the time between known availability and the transplant operation can be no more than 16 hours; ideally it is less than 10. Therefore, the service must operate 24 hours a day, 365 days a year. The computer centralizes tissue-typing information on all patients needing transplants in about 250 hospitals and compares it with similar data on the donor. The better the match, the less likelihood of rejection of the transplanted kidney. In less than 15 minutes from the time of the telephone call announcing an available donor, the computer lists the ten best matches nearby, notes the quality of each match, the degree of urgency on the part of the waiting patient, the blood group, and other relevant information. A national laboratory then chooses the recipient from that information and arranges to get the donated kidney to the waiting recipient.

Brain research is another area in which computers are used extensively to store and to analyze data. The complex mathematical calculations involved in the reading of brain wave patterns takes the computer only a few minutes to accomplish. It would take several years for a single human being to do the same thing.

Some psychiatrists are now even installing computers next to the couch! A computer program has been developed to collect patients' psychiatric histories, through questions displayed on a screen. Like the interviewing computer in the AMHT, this computer collects and analyzes much from the responses the patient types in. It even identifies those patients who are likely to attempt suicide.

## Government

At all levels of government, the computer helps to reduce bureaucratic inefficiency and save tax dollars. For example, in Nashville, Tennessee, computers plan more efficient routes for the city's garbage trucks, reducing the number of trucks and routes needed, saving time and half a million dollars a year. In Arcadia, California, a computerized master plan is planning for sewer requirements now, to prevent future overloaded sewers in this rapidly growing city.

In cities, computers are used to control traffic and transportation, collect and disseminate vast quantities of information, and help plan for the future. Computer simulations even help in designing altogether "new cities" of the future.

As society grows increasingly complex, it becomes impossible to govern ourselves efficiently without the help of computers. Various departments of government (like the state departments of motor vehicles, public utilities departments, social service agencies, education agencies, and the police departments) collect and store duplicate data. With proper use, the computer can (and in some states, does) centralize all records in a true "information system." The "National Data Center," which proposes such a system at a national level, would collect many of the overlapping files maintained by different federal agencies in a central computer system. The possible savings in cost and effort are obvious. However, whether Congress ever approves such a significant undertaking depends a great deal on whether acceptable guarantees for individual privacy can be worked out.

Politicians depend on computers to help them get elected. Computer simulations can forecast voter behavior in time for candidates to change their strategies before the election. Computers can predict the outcome of an election with great accuracy long before the time the final ballots have been counted.

And how are those ballots counted? By computer, of course. In most elections today, some kind of computerized voting system is used, to meet the demand for speedier returns. In the past, ballots were laboriously hand-counted, precinct by precinct, and the results of an election often were not known for days. Even then, the hand count was subject to errors, and recounts were sometimes demanded by defeated candidates.

Now, a "voting machine" might be used. Voters mark their X's in premarked boxes on paper ballots, which are automatically read and counted by an electronic mark reader. The results for an entire precinct are then recorded on a single punched card which is delivered to the central computer.

Another method uses a card as a ballot. Voters use a stylus to punch a hole corresponding to an X in a box for each candidate they wish to vote for. All the individually punched card ballots for each precinct are collected and delivered to the central computer for counting. Either method is less expensive than manual counting systems, and obviously faster and more error-free.

### Law enforcement

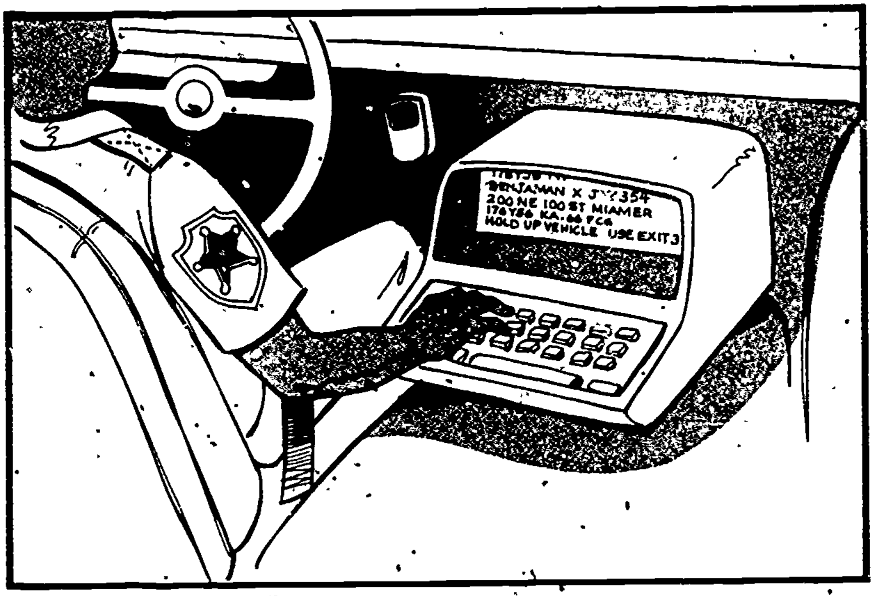
Automation in law enforcement is badly needed and is proceeding fast. In this country, serious crime is up 460 percent in the last 30 years, while the population has increased only 50 percent during that time. By 1980, 50 million Americans will have criminal records. The most effective method of preventing crime is the threat of rapid apprehension of the criminal. The computer is making it possible for immediate apprehension (through immediate information) to become a reality. Consider the following true-to-life incident:

> A police officer stops a car in a high-crime area at 4 A.M. The officer questions the driver, who behaves in a very nervous manner. The policeman is unsatisfied with the answers he gets, and suspicious because the address on the man's driver's license appears to have been altered. He radios his dispatcher at police headquarters for information on the man.
>
> The dispatcher faces a TV-type display terminal and keys in the following: "Johnson, William E., White male, DOB 10/3/39, resides at 416 S.W. Tyler St.". Immediately, data on four William Johnsons is flashed on the screen, but none of the birth dates or addresses match.

The officer returns to the car he has stopped and asks the driver for more identification. This time, the suspect comes up with a credit card from a local department store, with a different address than is shown on his driver's license. The new data is called in and the dispatcher sees that the address matches the third William Johnson shown on the screen. Immediately, the dispatcher requests the total record for this man and informs the officer that the man is wanted for robbery. From the time of the initial call to the moment the man was placed under arrest, 60 seconds had elapsed.

In Palm Beach County, Florida, sheriff's officers in 30 patrol cars have dashboard terminals with display screens. When stopping a car, the officers can get a "vehicle check" in less than 6 seconds by keying in the license number. Before leaving their cars the officers have complete information on the car and its registered owner. If there is any danger—if the car is stolen or the owner is wanted for a crime, for example—the officers will be forewarned. Through the same terminal, they have instant access to criminal and police records in the local

**Fig. 1–12** A computer vehicle check from a patrol-car terminal



computer, the state Crime Information Center, and the National Crime Information Center in Washington, D.C.

Even with police officers busy tracking down criminals, the average person driving to work still may not get away with exceeding the posted speed limit. A computerized, unmanned unit operating automatically day and night may catch the speeder anyway! This device, developed by an aerospace corporation, is set by a police officer for maximum and minimum speed limits. If a vehicle crossing a sensing device is traveling too fast—or even too slow—an automatic camera is triggered. The camera takes a clear photograph of the car and driver, and the computer superimposes the speed, location, time, date, and posted speed limit on the picture.

In spite of the tremendous increase in crime information, police departments continue to improve their speed and efficiency in using it. How? Through computerization of files and the instant retrieval of information from these files which is made possible. The "moniker" file is an example. Criminals often use a nickname or alias and a witness to a crime may overhear it. Individuals may then be identified by searching the computer file for all records which include that nickname. Then the description of the suspect is matched with the physical characteristics listed in the retrieved records.

The "M.O." file is another example. Many criminals follow a consistent method of operation, or "Modus Operandi." A burglar for instance, may always choose wealthy homes without servants, always between 2 and 4 A.M., always enter through a basement window, and always steal silverware but never jewelry. The computer could search the M.O. file to retrieve all criminal records showing this pattern.

Even the fingerprint file has been automated. The FBI file contains over 16 million different sets of criminal prints and 62 million more sets of civilian and military prints. Every day the FBI receives over 10,000 sets of fingerprints from arrests. The FBI must classify each set and search for a match. Because of the number of prints in the file, manual methods are very time-consuming. Computerization of this process contributes to speedier identification of suspects and apprehension of bungling burglars—and thus, hopefully, to the prevention of crime.

Investigations of crime activity can sometimes become boggled from an overload of information. The Senate Committee which investigated the "Watergate affair" finally resorted to computer, for this was the only way the mass of information could be handled. Once organized and stored in the computer, it was possible to retrieve the information alphabetically, chronologically, and by name and topic. The computer system responded swiftly to queries during the investigative hearings and then assisted committee members and attorneys in writing their report to Congress.

### Transportation

The airlines industry has become virtually dependent on computers. They not only handle ticket reservations, but help monitor and control air traffic in the air and on the airstrip, make up routes and schedules, assign crews and equipment, schedule regular maintenance on aircraft, and even handle and route baggage. As you know already from Millie's experiences, most airline companies have ticket reservation systems. These are designed to handle tens of thousands of passenger reservations a day. Information is usually stored in a very large central computer (in New York, for example) with remote terminals in airline offices all over the world. A reservation clerk needs only to push a few buttons on a console to complete a flight reservation.

In aerospace, even more astounding is the past and present research in airplane production. The structure and hence the operation of jet airliners has become so complex that a computer is needed just to navigate one into an airport. Pilots seldom single-handedly maneuver a large jet into a landing, because at some point the flight operation is set to computer control.

With the rising price of airplane production and fuel for flying, aerospace experts decided to cut costs if possible by finding more economical flight plans. They designed a flight plan system which could analyze the best course for each airplane to take, based upon current wind velocity and direction, temperature, and other factors. Since every five minutes of flying time costs about $150 in fuel alone, airline officials found the minutes saved made designing the system worthwhile.

While we might expect automation in an industry as complex as air transportation, it is sometimes surprising for an earthbound traveler to encounter a computerized transportation system while, say, driving to McDonald's for a hamburger. In our nation's capital, for example (and in several other cities), a computerized traffic control system similar to the one in Oklahoma City monitors the flow of traffic and automatically triggers the street lights at key intersections to provide the smoothest possible traffic flow. At the same time, 450 of the city's buses are equipped with radio transmitters to link the driver to the central traffic control computer. A driver may request a series of green lights by pressing a button. The computer scans the traffic in the area and, if the request is justified, sends commands to all the appropriate street lights. Such a system clearly could have implications for the environment—as soon as city dwellers learn that buses can speed them more quickly through city traffic, they may leave their cars in their garages and thus help to reduce pollution!

In England, the Department of the Environment has experimented with a driverless taxicab they call "Cabtrack." This four-seated vehicle is guided automatically along tracks. Passengers buy a magnetically encoded ticket at a cab stop and insert the ticket in the vehicle, which is then automatically directed to the required station.

If you decide to drive your own car, you may pull into a service station to find the friendly attendant replaced by a computer running the station on a 24-hour-a-day shift. You pull in, insert your credit card in the computer, get your gas, and receive a bill later (unless, of course, the credit card you insert has been listed as stolen or invalid—in which case the computer keeps it).

### Art and poetry

Although the computer cannot invent, create, or imagine, it can nevertheless be a useful and important tool for the creative artist. With

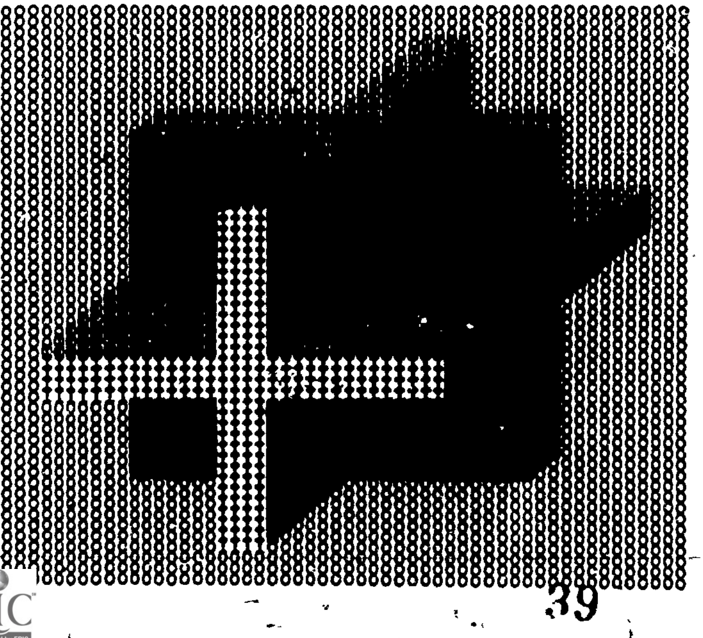


Fig. 1–13 Computer-assisted art

the help of the computer, artists are creating new and intricate works of art. Many of these have received the acclaim of critics.

Artists have long recognized mathematics as an important basis for artistic concepts such as repetition and proportion—two of the tools of computer art. A computer with a graphic-display-screen terminal can be programmed to generate, one after another, an endless series of mathematically graceful and pleasing forms. A person sitting at the terminal can become spellbound by the appealing and continually changing abstractions.

*Computers and People* magazine sponsors an annual Computer Art Exposition and publishes some of the best drawings. In Fig. 1-13 is an example from the August, 1973, Exposition which illustrates the variety of forms that is possible.

An artistic endeavor in which the computer has not been subject to as much critical acclaim or acceptance is the writing of poetry. You can see why, reading the computer-produced poem entitled "The Meditation of IBM 7094-7040 DCS":

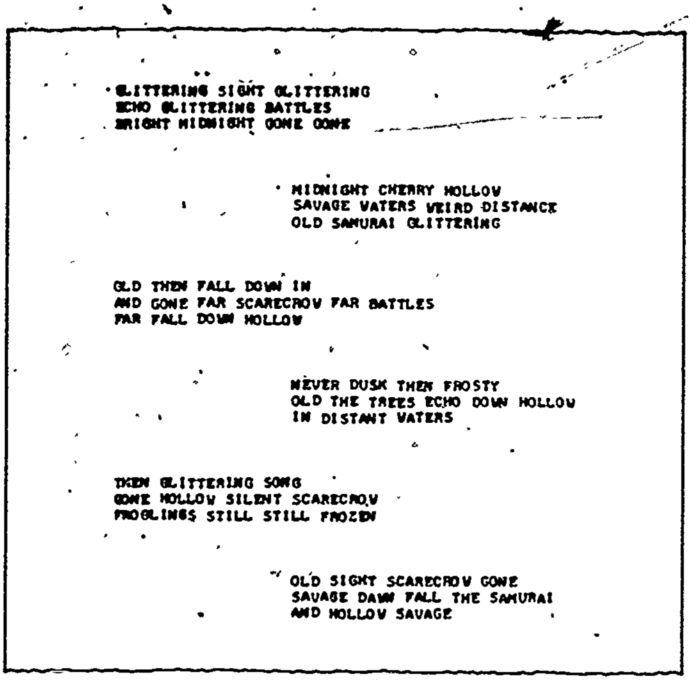
O poet,
Blush like a rotted skin;
Brighten like a dusty tower;
Wail like a happy earthworm;
Dream like an enormous flood;
Tremble like a red locomotive;
Flop like a damp gate! . . .
The beaches are praying.
Listen! How they stifle their
    enormous lips! . . .
The river
Winks
And I am ravished.[*]

Or you may find more meaning in one of those in Fig. 1-14.

## Recreation

Humans being human, it's not suprising that a major use of the computer is for fun. Many a blackjack or football game has been programmed for human vs. computer play. Chess-playing computers are no mere passing fancy—a computer vs. computer chess tournament was held in Boston in 1972 to determine the computer championship. Although the players were all rather bad, one has now been programmed to "learn" from bad moves or mistakes and will not repeat its mistakes. Scientists are placing bets now for the 1978 tournament.

[*] M. Borroff in "Programmed Poetry," *Time*, Feb. 22, 1971, p. 77.



```
GLITTERING SIGHT GLITTERING
ECHO GLITTERING BATTLES
BRIGHT MIDNIGHT GONE GONE


                MIDNIGHT CHERRY HOLLOV
                SAVAGE VATERS VEIRD DISTANCE
                OLD SAMURAI GLITTERING


OLD THEN FALL DOWN IN
AND GONE FAR SCARECROV FAR BATTLES
FAR FALL DOWN HOLLOW


                NEVER DUSK THEN FROSTY
                OLD THE TREES ECHO DOWN HOLLOW
                IN DISTANT VATERS


THEN GLITTERING SONG
GONE HOLLOW SILENT SCARECROV
FROGLINGS STILL STILL FROZEN


                OLD SIGHT SCARECROV GONE
                SAVAGE DAWN FALL THE SAMURAI
                AND HOLLOV SAVAGE
```

**Fig. 1–14** Computer haiku poems

At the 1972 Olympics, stopwatches and tape-toting officials were replaced by a computer. It determined placings, noted new records set, and transmitted these to scoreboards and remote display stations in the Olympic area and nearby Munich—all within 2 seconds after each contest had ended. In track, a light beam flagged the winner and flashed it to the computer. Swimmers touched electronic plates at the finish line. In field events, like the javelin or discus throw, a judge placed a prism at the point of impact, which was hit by an infrared beam. Distances were then measured by the reflected rays and sent to the computers via keyboard terminals. The computerized system was even used by doctors to analyze urine samples for traces of drugs. It was also used by attendants at the games to answer questions on Olympic history. A visitor typing in "Who won the 100-meter dash in 1896?" would get the computer response: "T. Burke of the United States in 12 seconds."

Auto racing also can be computerized. In one system, races are automatically scored by a computer. When a race car crosses sensors at the starting point of the track, a computer identifies it and starts timing, stopping when the car crosses additional sensors at the finish line.

Although—fortunately—you are not likely to find a computer in a remote campground, computers can help assure you one campsite will be empty when you get there! In some states, a computer-based ticket agency similar to the one Millie used to get play tickets will sell you tickets for your favorite public or private campground. The terminals are conveniently placed in pharmacies, department stores, or sporting goods stores. They will reserve your space, print your ticket, and even charge it to your credit card. The same computer can be programmed to assign campsites so as to give an area of the park a rest for grass to grow or to allow repairs to be made.. It can assign alternate, lesser known campgrounds to relieve the pressure on more popular ones, and can limit the number of hikers in an area that needs preserving. You can also ask for tickets for a river trip or a rock concert, rent a cabin in a campground, and reserve a motorhome, as well as a place to park it!

## AT EVERY TURN

There is virtually no area of human endeavor or interest which has not been influenced—or even vastly changed—by the computer. One of the more sophisticated computer users is the Department of Defense. At a quarter-million-acre weapons-testing military field laboratory on the California coast, an electro-magnetic environment has been created in which "nothing hostile can survive." Equipped with an awesome array of sensing devices, laser beams, night-seeing machines, and computers, this electronic battlefield ushers in a new philosophy of war which discounts the human factor. General William Westmoreland, while still the Army's Chief of Staff, forecast "combat areas under 24-hour surveillance, battlefields on which we can destroy anything we locate through instant communications and almost instantaneous application of firepower." • All of this brings to mind the science fiction image of machine facing enemy machine in a red blaze of gunfire and destruction, until finally the battlefield is quiet, littered with burned-out circuits and smoking electronic junk—an inhuman image, indeed. But where better to replace hnmans with machines than in a combat zone?

A very human use of computers is taking place in California, where a computer helps match homeless children and adoptive parents. The computer has been particularly helpful in finding good homes for such hard-to-place children as a deaf Japanese-American 5-year-old girl; two brothers and their sister who needed to be kept together; a blonde,

• Charles Foley, "Towards Computerized Warfare," *Current*, No. 132, Sept. 1971, p. 21.

blue-eyed boy, aged three, who was confined to a walker because of a birth defect.

After Joe Witless unlocks his apartment door at the Waterview Apartments in Framingham, Massachusetts, he is sometimes surprised by a voice over the intercom: "Your code, please?" Joe is forgetful. Unless he inserts his key in his "security control panel" to tell the resident computer he is home, the computer prints a warning to alert the security guard. The guard calls on the intercom, asking for Joe's code. If he does not reply with the correct series of numbers, the guard investigates the situation.

In this apartment complex the computer not only guards against unauthorized entry, but keeps watch for smoke or fire, and responds with prompt assistance if a resident pushes the "medical alert button" in his or her apartment. The computer can also increase elevator service during peak periods, ensure enough hot water for bathing, and even control the watering system for the 22 acres of lawn.

It is important to remember, as we encounter the computer at every turn, that all the marvels of the computer revolution were first imagined in the mind of man or woman. Before the computer can be put to work in any of the marvelous ways described earlier, it must be programmed by a human being. It must be told exactly, in minute detail, how to perform each part of every task. If the computer performs brilliantly, it is because it has been programmed brilliantly. If it goofs, it is because the programmer goofed. A familiar computer adage says this in a more abrupt way: "Garbage in, garbage out." As dependent on the computer as we may become, the computer is likewise obliged to remain totally dependent on humans to tell it how to function.

Because computers can relieve humans of the necessity to do routine mental tasks, we find ourselves with more freedom to develop our creativity. While computers carry on the more mundane work of society, humans can organize, invent, act on a hunch, create, and wonder—and put the computer to work producing the inventions, studying the hunch, and helping to discover the answers.

In your lifetime, you will learn to use and to live with computers. Whether or not you choose a career in which you deal with them directly, you cannot escape their influence. In the world of the future you may choose to be an informed "consumer" of computer goods and services, or you may choose to be one of those who make it happen—a computer operator, a data processing clerk, a systems analyst, a programmer, or perhaps a computer salesperson, engineer, or maintenance person. This book is intended to give you some experience with and understanding of the computer, and of computer or computer-related careers. After this exposure, you should have a better idea of your own interest in the field, and which, if any, computer career you wish to explore in more depth.

If you decide not to go further in the computer world, you still will be ahead. You will have had enough experience with the computer to be able to make intelligent use of it in whatever career field you enter. Like death and taxes, the computer seems to be a "sure thing." You will probably encounter it in use in some way whether you become a librarian or a firefighter, a carpenter or a file clerk, or a fashion designer. The more you know about using a computer, the better equipped you will be for your chosen career. And you get a bonus—you are becoming an "informed citizen" where computers are concerned. You will not be an ignorant consumer, accepting anything that comes from computers without question or judgment. You will be one of those valuable citizens who deal with the computer intelligently and usefully. You will neither reject it out of ignorance nor apply it like a Band-Aid to every situation. You will use it instead for what it is best suited—extending the abilities of human beings.

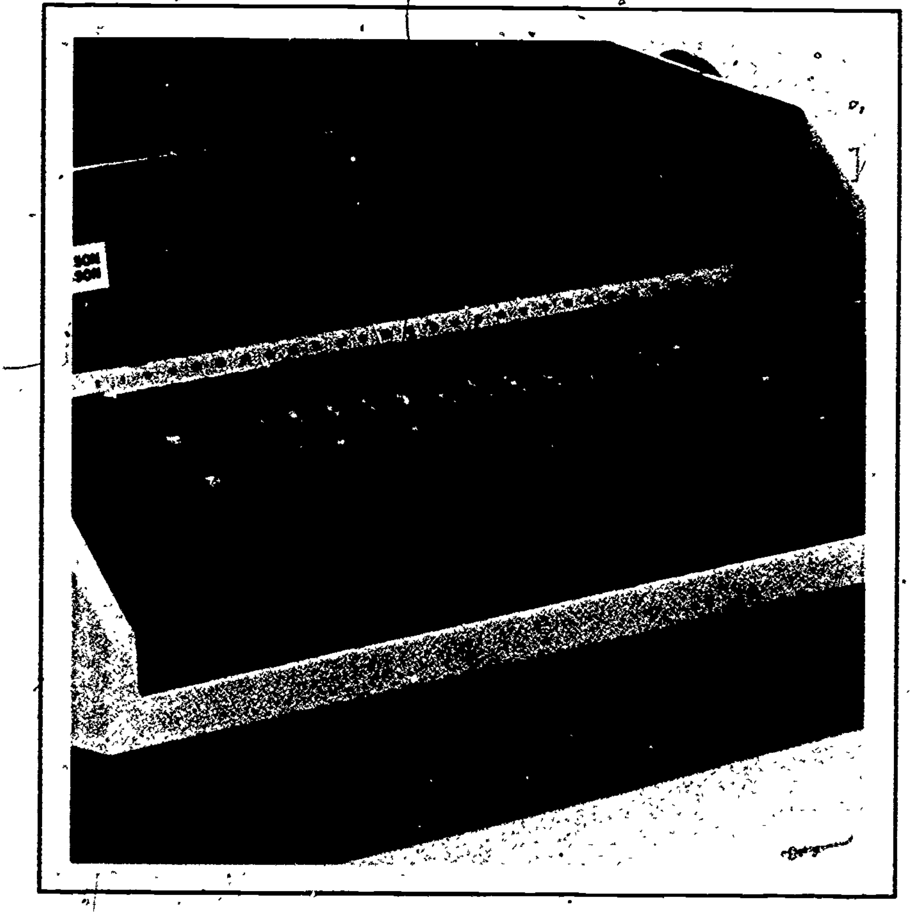
## Check your understanding

1. You may have heard of "computer dating services" which match up people according to interests, backgrounds, physical characteristics, likes, and dislikes. Does this seem to you to be a good way to meet potentially compatible people? Would you subscribe to such a service? Why or why not?

   Would you subscribe to a computer service which selects a spouse for you? Why or why not? What elements (if any) in the process of "selecting a spouse" could not be quantified?

   Argue or debate this question with a classmate, taking the position that "better and longer lasting marriages would result if computers matched people for marriage." Then switch sides and argue the opposing point of view.
2. List examples of problems or decisions that *cannot* be handled by a computer. Then list problems that *should not* be handled by a computer. Are the lists different? In what ways? Discuss with your classmates the reasons for the differences.

# 1234567



# The nature of the computer system

## INPUT, PROCESSING, AND OUTPUT

When you solve a problem of any kind, there are usually three operations involved. First, you receive some *input* (you read or hear information about the problem), then you *process* the information (solve the problem), and finally you *output* the results.

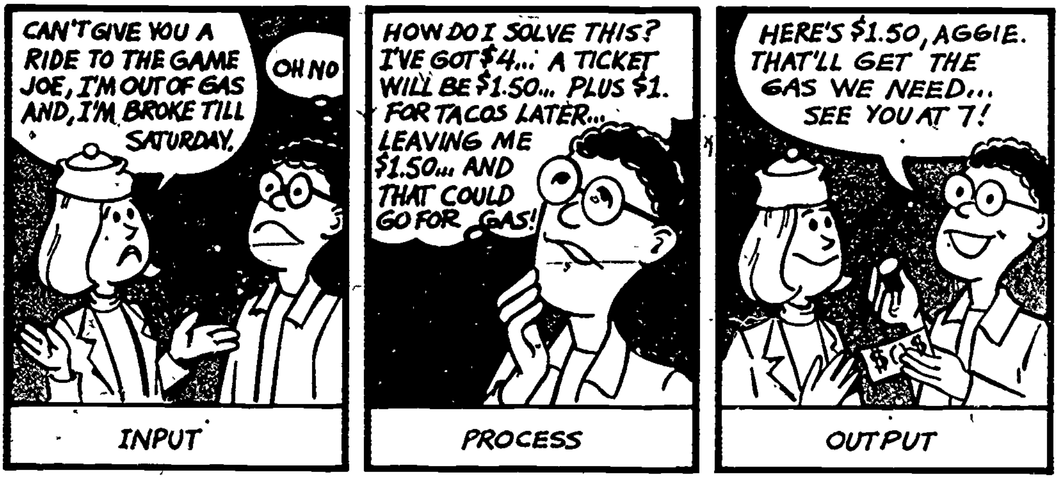
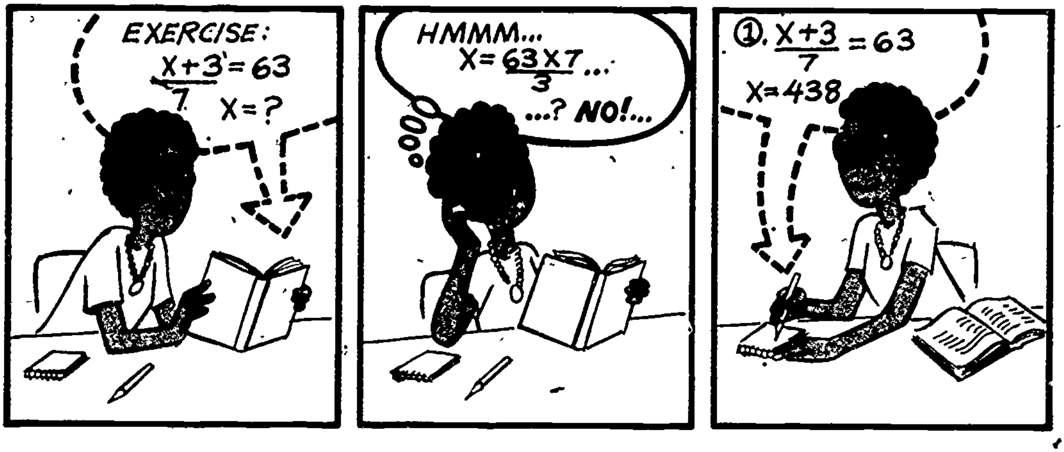

CAN'T GIVE YOU A RIDE TO THE GAME JOE, I'M OUT OF GAS AND, I'M BROKE TILL SATURDAY.
OH NO
HOW DO I SOLVE THIS? I'VE GOT $4... A TICKET WILL BE $1.50... PLUS $1. FOR TACOS LATER... LEAVING ME $1.50... AND THAT COULD GO FOR GAS!
HERE'S $1.50, AGGIE. THAT'LL GET THE GAS WE NEED... SEE YOU AT 7!
INPUT
PROCESS
OUTPUT

Fig. 2–1 Example of input, process, and output

Here's an example. Suppose a student plans to get a ride to the game with another student. But, a problem comes up. The student with the car is almost out of gas and has no money for more. Here are the three steps involved in solving the problem.

Or, consider a girl working a math problem. She reads the problem in the book (input), solves it in her head or on paper (process), and writes down the answers on her paper (output).

Fig. 2–2 Example of input, process, and output



Here's a problem for you to solve:

On a separate piece of paper, write the states for each of the capital cities listed below.

*Capital City*

Sacramento  
Atlanta  
Trenton  
Cheyenne  
Honolulu

Now think about all the operations involved in solving that problem. Can you identify the input, process, and output operations? A description of the three operations for solving this problem is found below. Note them on your paper. Then label each operation as an input, output, or process operation.

*Operation Description*

1. Recall the names of the appropriate states, or look them up, or ask a friend.
2. Read the problem and the list of capital cities.
3. Write the names of the states.

A human being solving a problem is one example of a "system" in operation, going through the input-process-output cycle.

Many other systems follow this same input-process-output sequence. Consider a clerical system—for example, the system in which class schedules are set up. When you hand in the list of classes you want to take next term, that's input. The school office gathers all the students' class lists together and assigns classes to each student—that's processing. Finally, the office issues each student his or her class schedule—that's output.

Or consider a mechanical system. When you put a quarter in the soft-drink machine and push the "Coke" button, the machine selects a bottle of Coke, drops it into the tray, and delivers your change. Which operations are input, processing, and output?

Just as these systems operate in an input-process-output sequence, so does a computer system. Data are input to the computer from a user, the computer processes that data and then the computer outputs the resulting information.

Do you have input, processing, and output clearly in mind? See if you can answer the following questions.

## Check your understanding

1. If you used a desk calculator to add 4 and 5, what would the output be?

2. Suppose you use a computer to figure out the Grade Point Averages (G.P.A.'s) for everyone in your class last term. Three things would be involved: 1) the calculations of G.P.A.'s, 2) the printed list of G.P.A.'s, and 3) the original grades for each student. Which would be the input operation? The processing operation? The output operation?

3. a. You insert a Canadian quarter in an American soft drink machine and push the "orange" button. The machine returns your quarter and no orange drink. Was the error made in input, in processing, or in output?
   b. You retrieve the Canadian quarter, use an American coin, and again push "orange." The machine drops a cup into place and begins filling it with orange drink, which knocks the cup over and spills it. Was the problem in input, processing, or output?
   c. You try again. This time you push the "cola" button. The machine gives you a cup of grape drink. Was the error in input, processing, or output?
   (Time to give up and find the drinking fountain!)

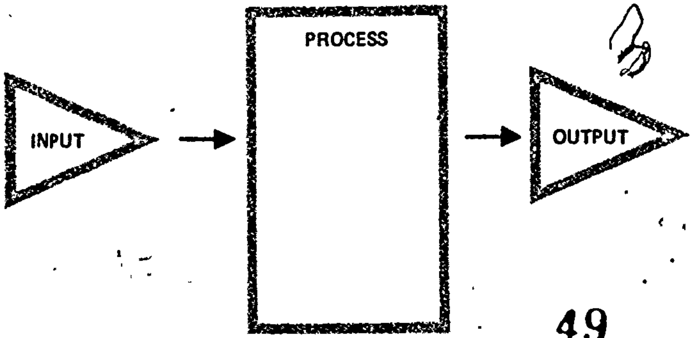


**Fig. 2–3** The three operations in a computer system

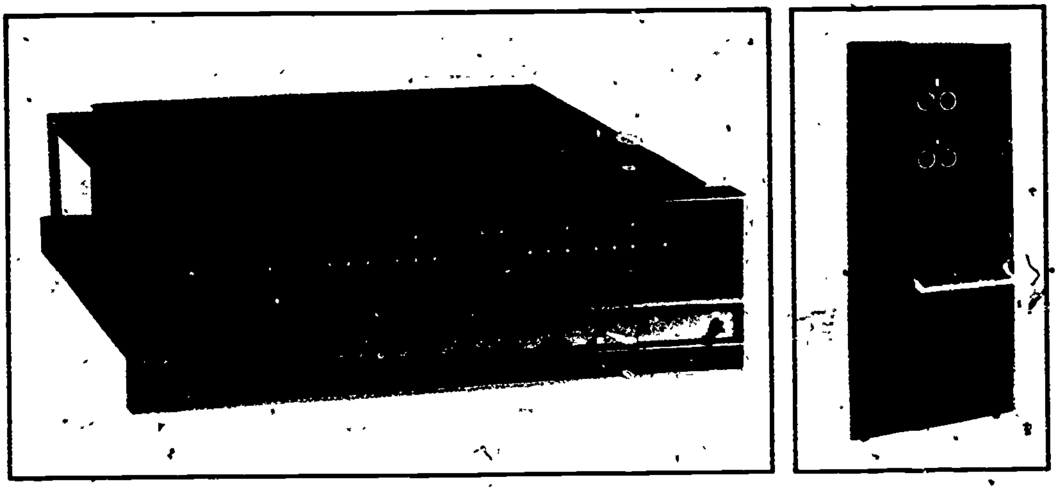


Fig. 2-4 Central processing units come in all sizes. Nova 1210 Mini-computer (left) and Digital POP-12 computer (right) which is many times larger

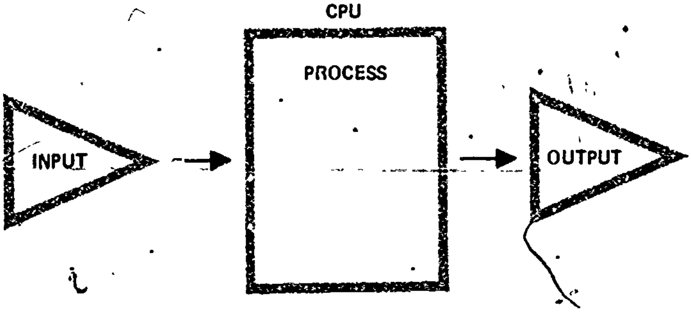
## THE THREE PARTS OF A COMPUTER SYSTEM

Let us take a closer look at the parts of a computer system, as we have diagrammed it in Fig. 2-3.

How does a computer "process" information? What piece of equipment is responsible for this "processing"? The answer is the heart of every computer system is a single piece of equipment, sometimes as small as a breadbox and sometimes as large as a refrigerator or larger. It is called the central processing unit or the CPU. Two different CPU's are depicted in Fig. 2-4.

The computer equipment cannot, however, process information by itself. A computer is a totally witless machine. It must be given instructions, telling it exactly how to solve any problem, in step-by-step detail. These sets of instructions, prepared by human beings, are called *programs*. The programs are as essential as the machinery in processing data.

Fig. 2-5 Input, CPU processing, and output



The computer, being a mechanical, electronic device, needs some mechanical or electronic means of storing information—both programs and data. When you processed information to write down the state for each capital city (on page 41), you had to retrieve some information

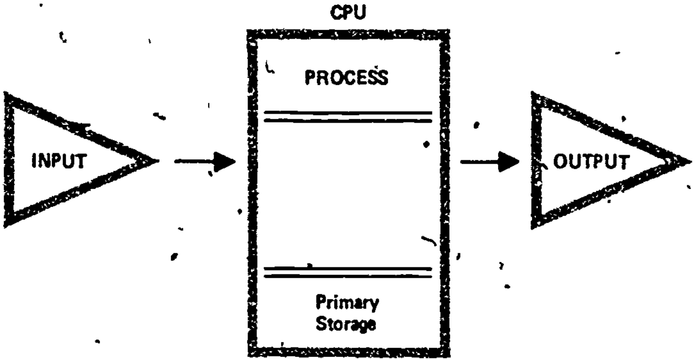


**Fig. 2–6** Primary storage in the CPU

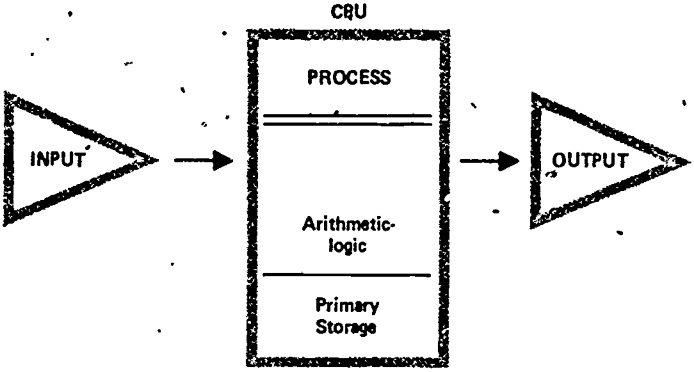


**Fig. 2–7** Arithmetic-logic unit in the CPU

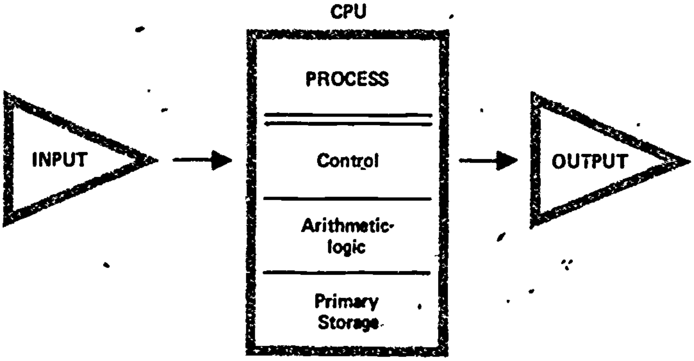


**Fig. 2–8** Control unit in the CPU

you previously had stored somewhere in your memory. Similarly, the computer needs some place to store information and to store programs. Some of this storage is found right in the CPU. It is called "primary storage."

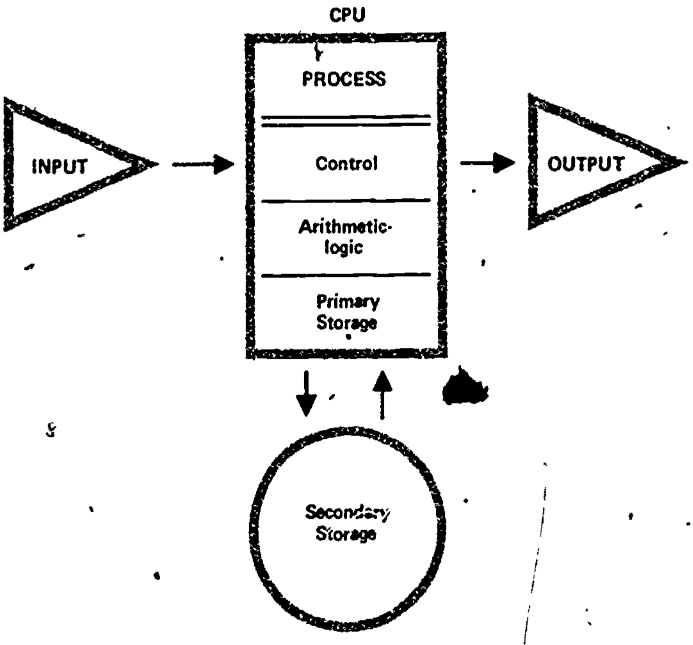
The CPU also must be able to do arithmetic and to manipulate words, letters, numbers, and other symbols. This symbol manipulation is done in the "arithmetic-logic unit" of the CPU.

Finally, all of the processing must proceed in an orderly manner. Part of the CPU is devoted to overseeing the processing operation. This part controls the processing, to ensure that it takes place in the proper sequence. It is called the "control unit."

Remember again how you processed information to write down the state for each capital city. Perhaps you hadn't previously stored the names of all 50 U.S. capital cities in your memory. In that case, you may have had to ask a friend for help, or even look the names up in an atlas or geography book. If you did, you were using "secondary" storage.

The primary storage in the CPU is usually not large enough to store all of the programs and data a computer needs to do all the processing that people require it to do. So, most computer systems have a great deal of additional storage available outside the CPU, in other pieces of equipment. This outside storage is called "secondary storage," and is shown in Fig. 2-9.

Fig. 2-9 Secondary storage available to the CPU



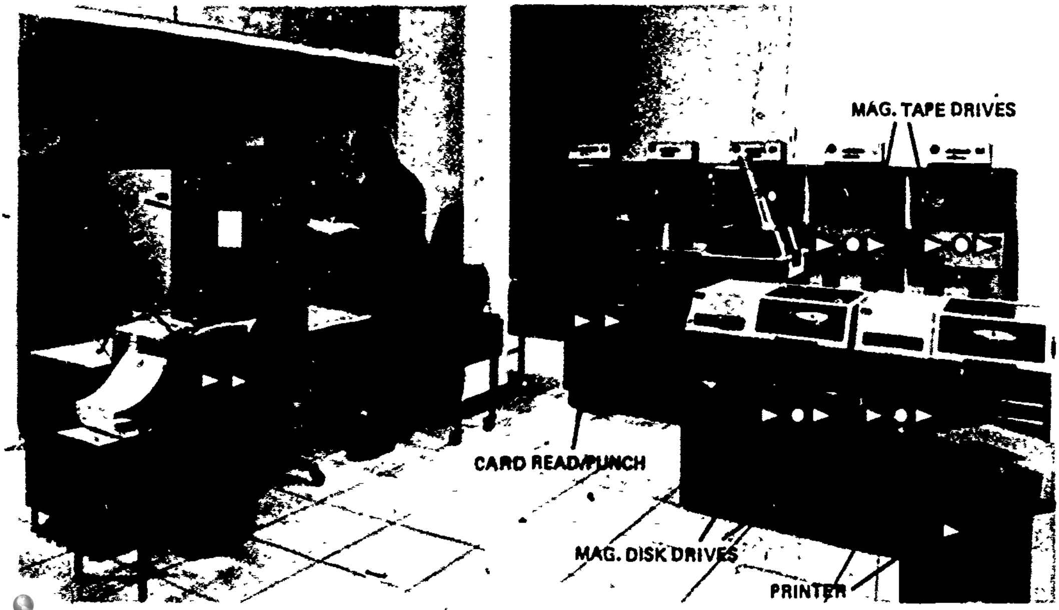
Now you know all of the parts of a computer system. The pieces of equipment you see in a computer room are devices used for *input* of information, for *processing*, or for *output* of results. Some devices are used for *secondary storage* of information needed by the computer.

Here is a picture of a complete computer system, with symbols showing just which function is performed by each device. Notice that, as is common in most systems, a punched card reader is used for input and a line printer for output.

As you can see from the symbols, some devices perform two or three operations. In a combined card reader/punch unit, for example. the card reader can be used as an input device (to read previously punched cards) or the card punch can be used as an output device (to punch holes in cards). Which other devices shown in Fig. 2-10 operate both to input and to output information? Which are also used for secondary storage?

To get a more concrete idea of what is actually involved in the input-process-output operation, consider the simple example of processing some students' names to get a list of the names in alphabetical order. First, a program telling the computer how to alphabetize is loaded into the computer. Then, put in the students' names and get the alphabetized list.

Fig. 2-10 Complete IBM 360 computer system



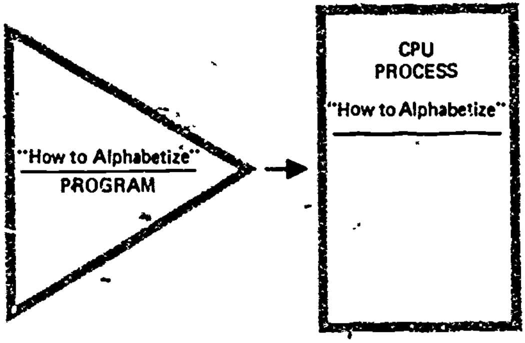


Fig. 2-11 Loading the program

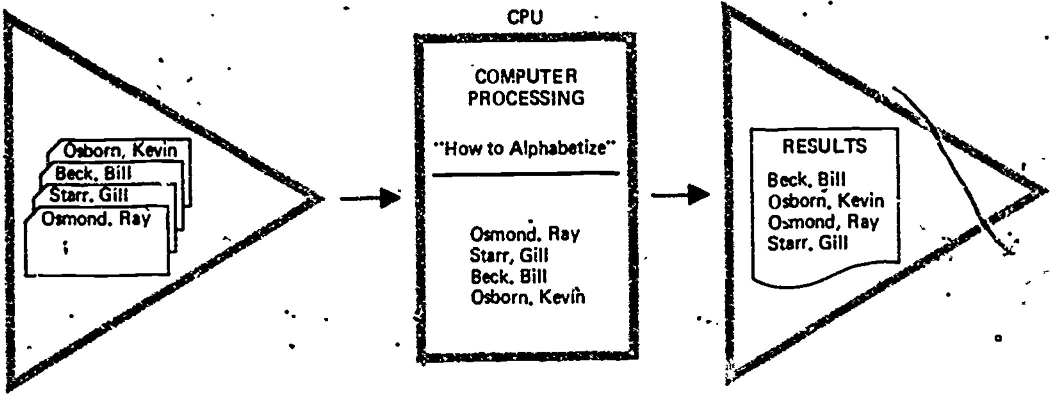


Fig. 2-12 Processing the data

In every case, data processing involves the input of some original data, its processing by the computer, and the output of new, usable information. This is true whether student grades are being input to get an honor roll list or measurements are being input from instruments in a moon capsule to get an analysis of its flight condition. And, in every case, computer operators oversee the actual processing of the data.

So far, we have considered the computer operation sequence as a three-step sequence: input-process-output. As you may have suspected, however, there is often an additional step involved in a data processing cycle. Since computers can't read books or listen to instructions, the input has to be in a form the computer system can interpret. For example, the programs or data could be read into the computer from coded holes punched in a card or magnetized spots on a tape. If it isn't already in such a form (such as holes on cards or magnetic spots on a

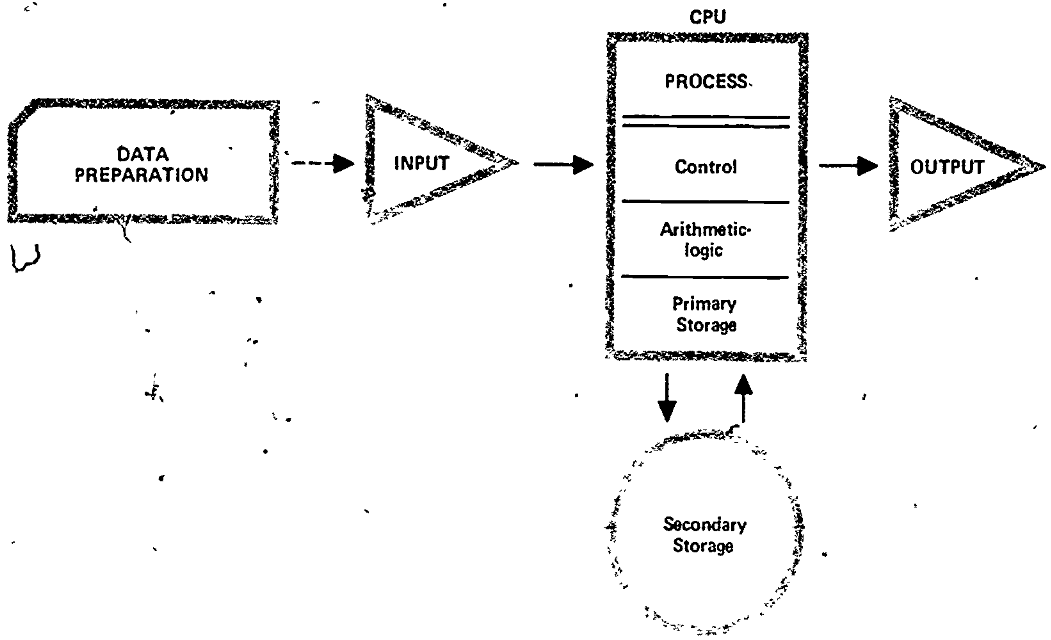


Fig. 2-13 Data preparation: an additional step

tape), it will have to be put in an appropriate form. This would add a necessary step to the sequence: data preparation.

You will sometimes find in a computer room pieces of equipment that are used solely for data preparation. Keypunch machines, for example, have only one use—to punch coded holes in the cards which will later be read into the computer as input.

Later in the chapter, you will be learning more about all of these devices—machines for data preparation as well as machines for input, storage, processing, or output.

Before going on to look at the way one computer can be used by many people, answer the following questions to see if you have a clear understanding of the things you have read so far.

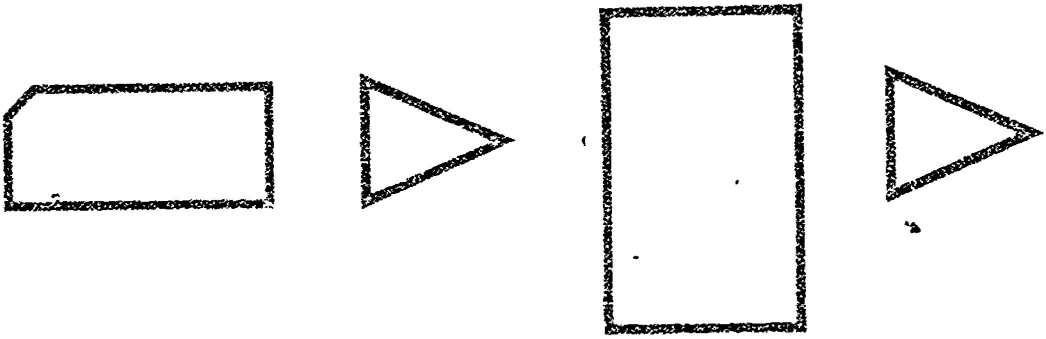
## Check your understanding

1. What is the name for the part of the computer system which carries out the processing?
2. What is a computer program?

3. Which of the following are parts of the CPU?

   a. Control unit
   b. Data preparation machine
   c. Primary storage
   d. Input device
   e. Arithmetic-logic unit

4. When you are asked for a person's phone number and you have to look it up in your phone book, which kind of storage are you using—primary or secondary?
5. Name a common input device.
   Name a common output device.
6. Show the four steps in the computer processing sequence by copying the diagram below and filling in your diagram with labels and arrows for each step.



7. In your diagram, draw the symbol we are using for secondary storage in the correct place, label it, and show by arrows how it interrelates with the other parts of the diagram.
8. Why is the first step in the processing sequence needed?

## ONE COMPUTER—MANY USES

During the first dozen years in which computers were used, the pieces of equipment that made up the computer system were usually found all together in one central place, as shown in Fig. 2-14. Anyone who wanted to use the computer had to prepare the data and bring or mail them to the computer center, where input devices would read the data directly into the central processing unit (CPU) and where output devices would print the results.

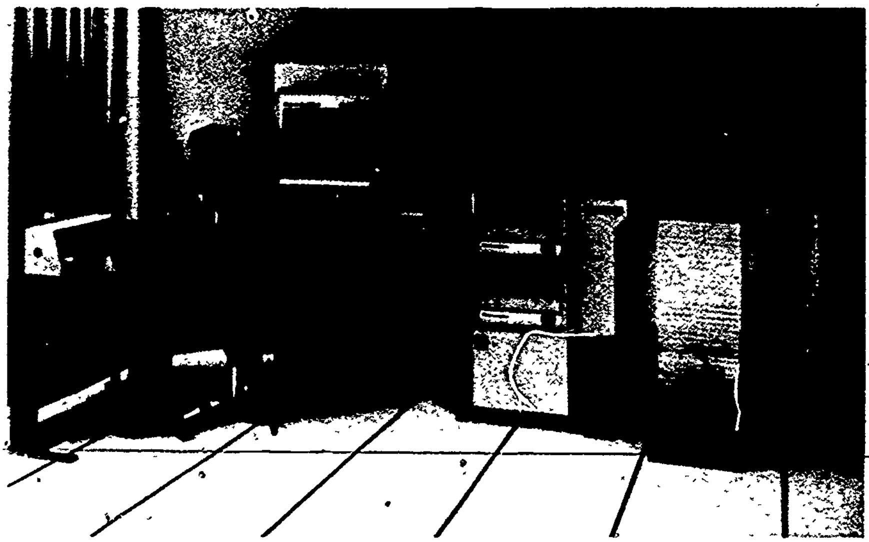


**Fig. 2–14** A Burroughs B1700 computer system in one central place

**Fig. 2–15** Time-sharing terminals connected to a distant CPU

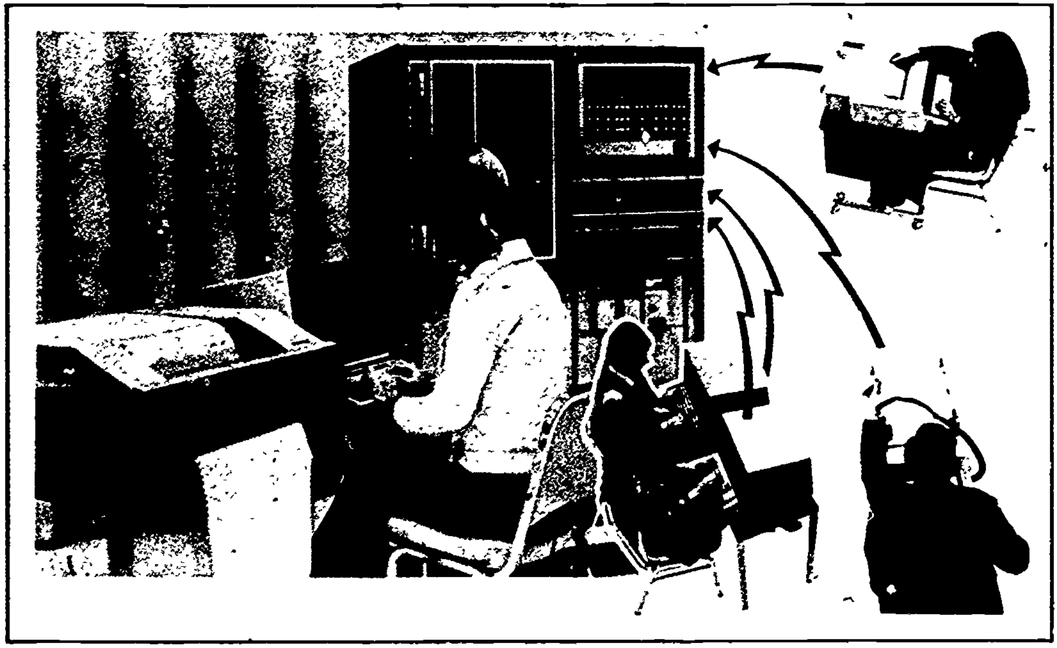


Fig. 2–16 Time-sharing terminals can be located anywhere.

In recent years, a new method of using the computer called "time-sharing" has become popular. It makes the computer more convenient for many people to use. Time-sharing computer systems still have all the usual pieces of equipment but, in addition, they have input/output devices called *terminals* which can be located in the next room, the next block, the next town, or thousands of miles away. The terminals can be connected to the central computer by ordinary telephone lines. With time-sharing, users can input their data and immediately receive the output right in their offices, at a department store check-out counter, in the warehouse, the classroom, or even in their homes.

Computer time-sharing makes it possible for anyone, anywhere, to have direct and immediate access to a computer.

Whether the time-sharing user is at the computer center or at a terminal, the central computer system he or she uses will have the components shown in Fig. 2-17.

But, in this case, some of the input/output devices (terminals) will be located at a distance from the main computer system.

Usually, the terminals in a time-sharing computer system are simply electric typewriters. The user types on the keyboard, and the information he or she types is transmitted over a telephone line as *input* to the computer. When the computer has information to *output* to that user, the information is transmitted from the computer over a telephone line, and then it is automatically printed out on the user's typewriter terminal.

Fig. 2-19 is an example of input and output from one terminal. The program telling the computer how to process this user's input had been stored in the system at an earlier time. In this example, everything typed by the user—all input—is in lower case. All output from the computer is in upper case (capital letters).

As you can see, the person at the terminal can have a "conversation" with the computer. This kind of "conversational" use of the computer (also called *interactive processing*) is one advantage of using a time-sharing system. Each time the user enters input, the computer immediately responds with output. This is *not* the way a conventional computer system, without terminals, handles input and output. In a conventional system, all input must be provided in one batch, and there is usually a wait for the output. The wait may be only a few minutes, or it may be as long as a day or two.

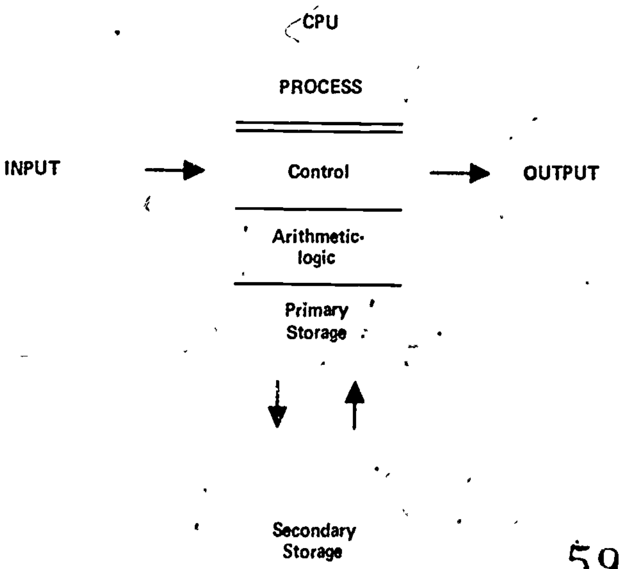


**Fig. 2-17** Components of a time-sharing computer system

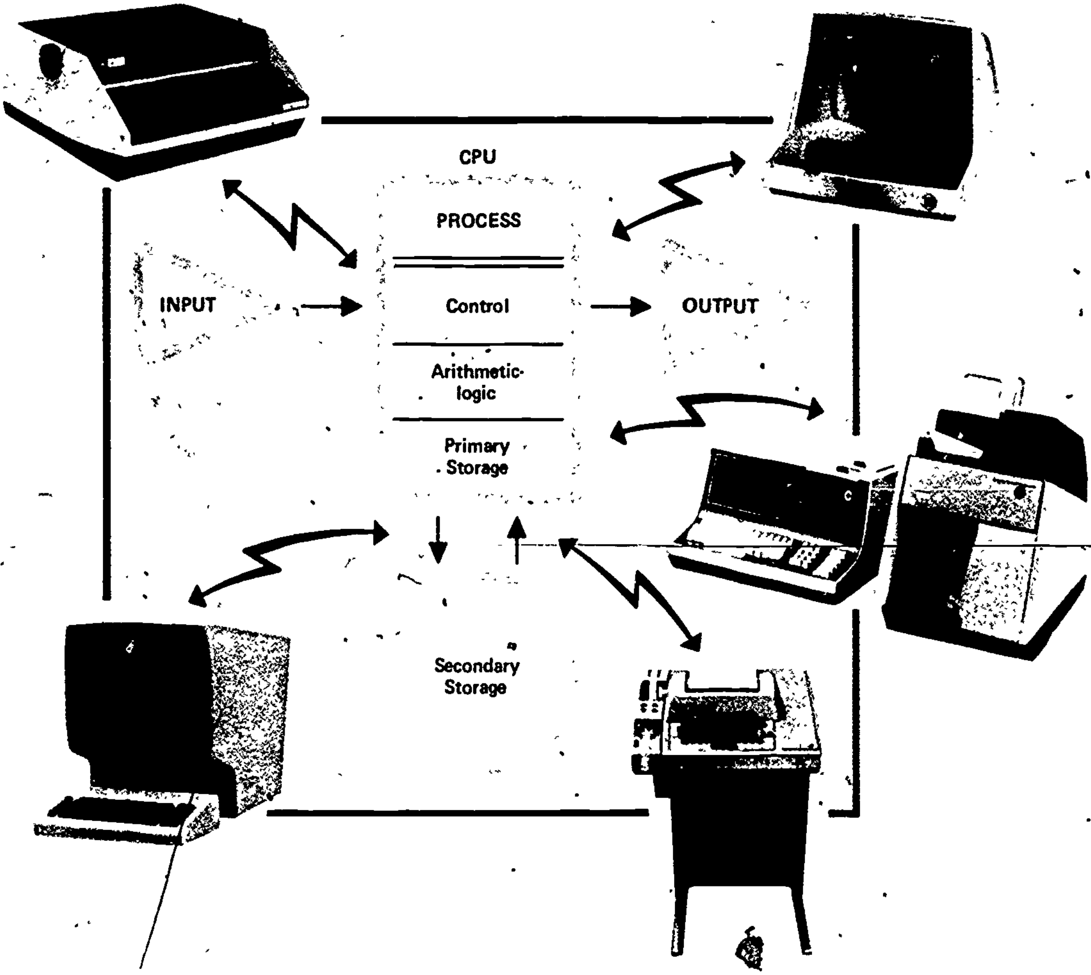
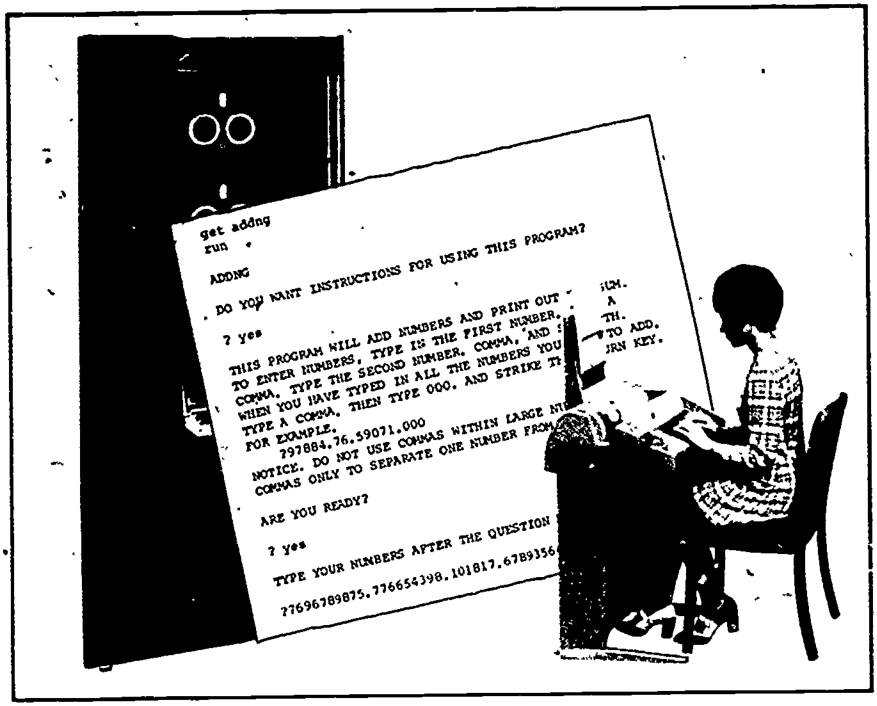


**Fig. 2–18** Terminals connected to the time-sharing system*

Why is there a wait for output in a conventional system? For the sake of efficiency! It is more efficient to collect a lot of input data of the same type and to enter all of this input data in one *batch*. This is called batch processing. It is characterized by "batching" or grouping transactions so that as much work as possible will be handled all at one time. A good example of this can be seen in a large city water department. Each customer's account normally has only two transactions a month. One bill is calculated and one payment is posted. It is not worthwhile to

* For communication between a terminal and a computer or between computers, some additional equipment is involved. Computer users will be introduced to that equipment in the operating manuals for terminals and computers.



```
get addng
run

ADDNG

DO YOU WANT INSTRUCTIONS FOR USING THIS PROGRAM?

? yes

THIS PROGRAM WILL ADD NUMBERS AND PRINT OUT      SUM.
TO ENTER NUMBERS, TYPE IN THE FIRST NUMBER,      A
COMMA, TYPE THE SECOND NUMBER, COMMA, AND S      TH.
WHEN YOU HAVE TYPED IN ALL THE NUMBERS YOU       TO ADD.
TYPE A COMMA, THEN TYPE 000, AND STRIKE TH       JRN KEY.
FOR EXAMPLE.
     797884.76.59071.000
NOTICE. DO NOT USE COMMAS WITHIN LARGE NU
COMMAS ONLY TO SEPARATE ONE NUMBER FROM

ARE YOU READY?

? yes

TYPE YOUR NUMBERS AFTER THE QUESTION

77696789875.776654398.101817.6789356
```

**Fig. 2–19** Input and output with a time-sharing system

process each payment immediately each time a transaction occurs. Instead, billing is done in one batch for all customers once a month. Payments come in on a random schedule through the month, and a card is punched for each payment as it is received and deposited. These cards are collected, or batched, and processed as a batch once a day.

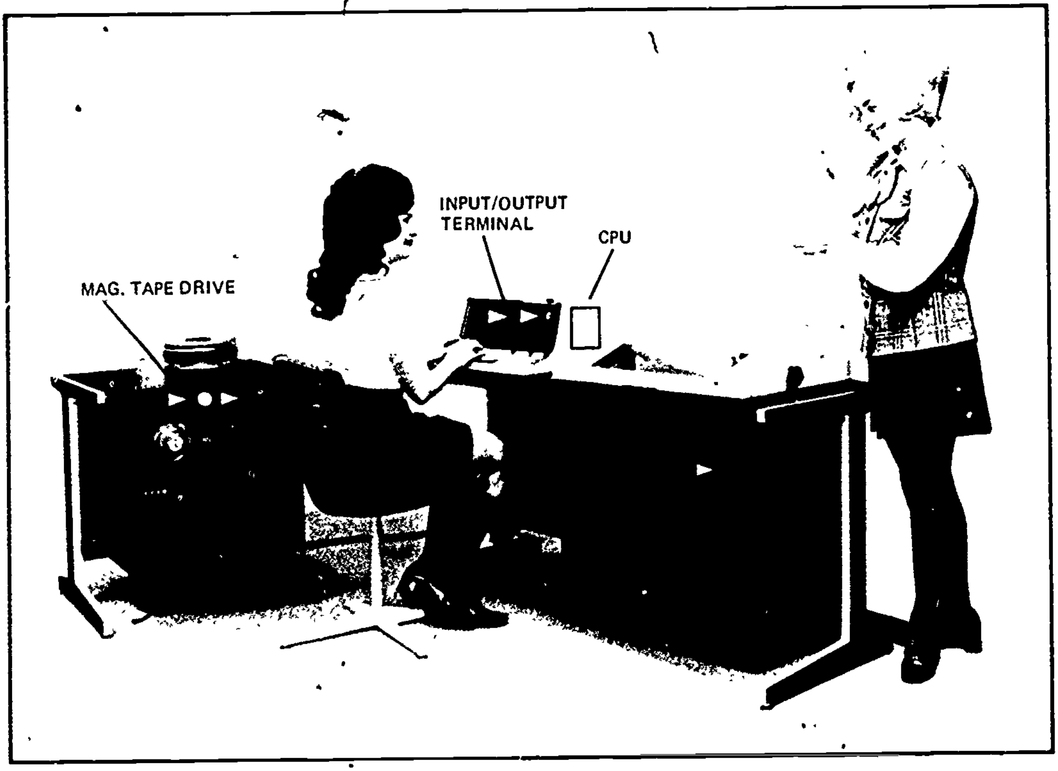
The two types of processing, then, ~~are~~ called interactive processing and batch processing. Which one is associated with time-sharing? Which one with conventional computer systems? Which one is most likely to require the "data preparation" step? Which one is probably the most economical way to process data?

Interestingly, many time-sharing computer systems which handle remote terminals usually can do batch processing at the same time. One part of the system controls input and output to the terminals while another part continues to accept and process batches of data in the usual way. Look at the diagram in Fig. 2-18. You can see that the CPU is diagrammed as processing input from terminals at a distance as well as from an input source located right next to it. Do you see one reason why such a system is called "time-sharing"? Both the interactive and batch processing usually is done in the same CPU. Of course, this requires some very clever scheduling and juggling of data to avoid mix-ups. This scheduling is handled by special programs which keep everything operating efficiently.

When time-sharing systems were first introduced in the 1960's, they were seen as revolutionary. Today we are seeing a revolution in terminals. For years, remote terminals connected to a time-sharing system were nearly always simple electric typewriters, used for input and output. But today, a "terminal" may take many different forms and is no longer so simple. Improved models of electric typewriters and CRT (cathode ray tube) terminals are commonly in use today. In addition, a new and highly versatile type of terminal has been developed, called an "intelligent terminal."

An intelligent terminal is more than one piece of machinery and does more than double duty. It combines a small CPU of its own (a "mini-computer"), one or more input devices, one or more output de-

**Fig. 2–20** Example of an intelligent terminal



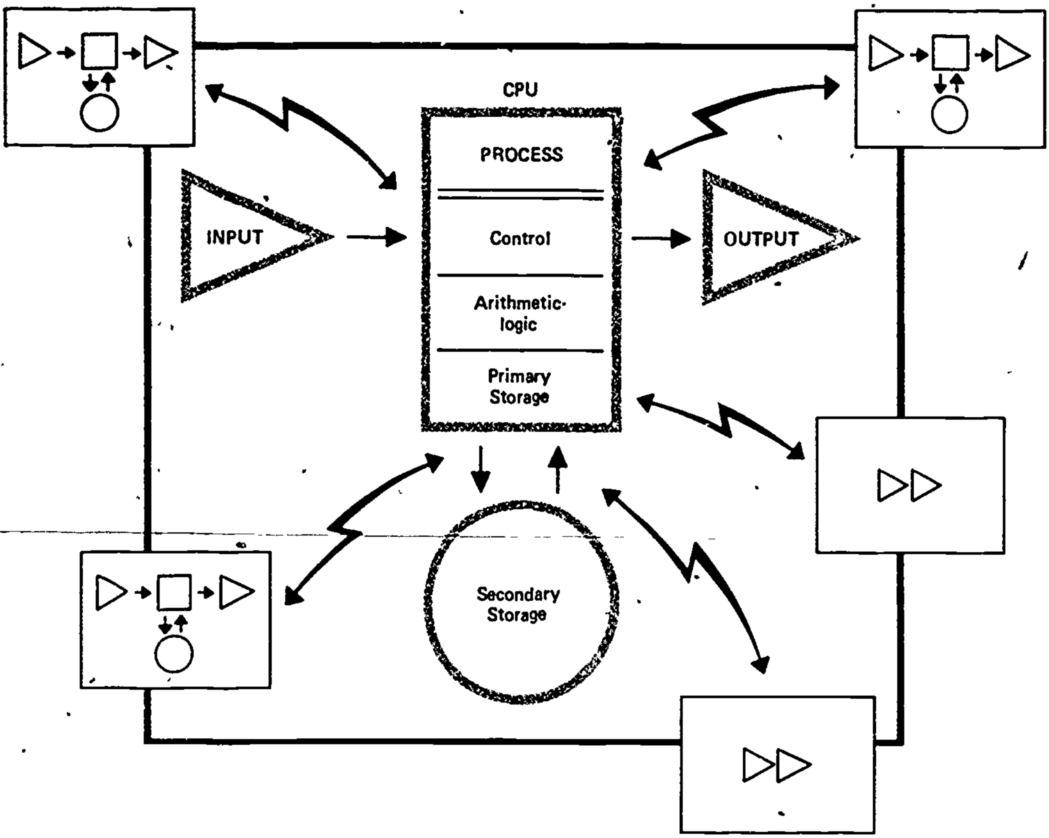


**Fig. 2-21** Central computer system with ordinary terminals and intelligent terminals connected to it

vices, and sometimes a secondary storage device or two. It can be used as a remote terminal connected to a very large time-shared computer by telephone line. It can also be used all by itself to process data in its own small CPU and to solve problems that aren't too lengthy or complex. It can store and use only short, simple programs. To solve more sophisticated problems or to process large volumes of data, it will again act as an input/output device to the larger computer.

An intelligent terminal will have, at least, a keyboard terminal (possibly with a display screen like a small TV screen called a cathode ray tube, or CRT), and a small CPU capable of storing some programs in its primary storage unit. In addition, it may have one or more conventional electric typewriter input/output devices and various other input, output, and secondary storage devices.

Fig. 2-21 is a diagram of a computer system with intelligent terminals and regular terminals.

Now check your understanding of what you have read so far by answering the following questions.

## Check your understanding

1. Which of the following kinds of computer systems does your school district use? (Ask someone who knows.)
   Conventional batch processing computer system.
   Time-sharing computer system
   Combination batch and time-sharing system
2. Ordinary terminals usually include which of the following devices?
   a. CPU
   b. Input keyboard
   c. Secondary storage
   d. Output device
   e. Primary storage
3. Which of the following is true of an ordinary terminal?
   a. Can process limited amounts of data in its own processor
   b. Can input and output information to and from a central computer
   c. Can store data
4. In Fig. 2-16, what clue can you find in each picture of a terminal to indicate it is connected to a time-sharing computer?
   a. Human operator
   b. Keyboard
   c. Telephone
   d. CPU
   e. CRT
5. For the user, an important advantage in using a time-sharing system is
   a. talking on the telephone.
   b. processing an entire batch of information all at one time.
   c. it keeps everything from happening at once.
   d. fast, interactive processing.
6. *Interactive processing* means
   a. immediate response from the computer each time the user enters input.
   b. conversational use of the computer.
   c. neither a nor b.
   d. both a and b.

7. The most important distinction between interactive processing and batch processing is

    a. batch processing is faster.
    b. interactive processing provides an immediate response to input.
    c. batch processing is "conversational" in nature.
    d. in interactive processing, input data of the same type is collected for entering in a single interaction.

8. Which is true of intelligent terminals?

    a. Can process limited amounts of data in its own processor
    b. Can input and output information to and from a central computer
    c. Can store data
    d. All of the above

9. All intelligent terminals include which two devices listed below?

    a. CPU and secondary storage
    b. Keyboard and line printer
    c. CPU and keyboard
    d. CRT and secondary storage

10. What kind of terminals are available in your school district?

    a. Electric typewriter (or teletypewriter)
    b. Display screen with keyboard (CRT)
    c. Intelligent terminal
    d. Other (describe)

11. If you have an intelligent terminal, answer these questions.

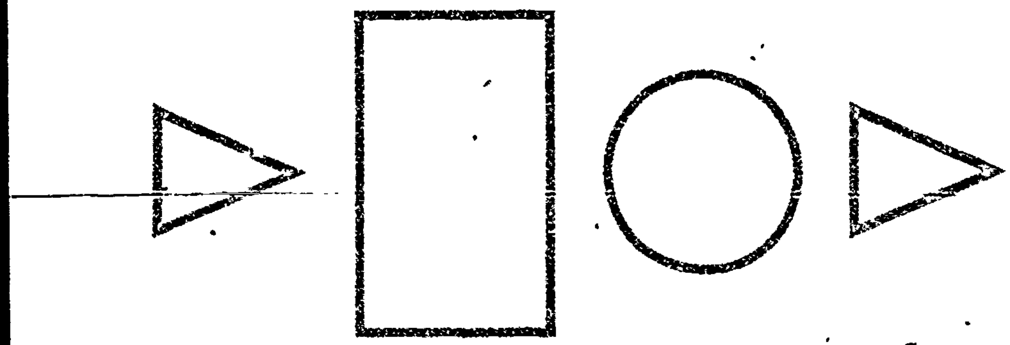
    a. Does your intelligent terminal have secondary storage?
    b. What kind?
    c. Does it have more than one input/output terminal?

## COMPUTER CODES FOR HUMAN DATA

In a human "data processing system" (for example, a person who recalls and lists states to match their capital cities) input is straightforward and simple. Human beings can receive input through their senses—sight, hearing, touch, and so on—and can understand spoken or written words and sentences. Human beings understand a vast number of symbols. Try to count all the symbols you understand. Here are a few:

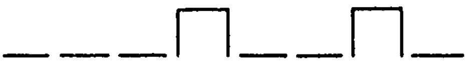
A, B, C, 8, 4, 3, ?, +, =, :, ", →, !, 🯅, $

What's more, human beings can learn the meaning of new symbols.. Can you write the meaning of each of the computer operations symbols below? They are symbols for which you have learned a new meaning.

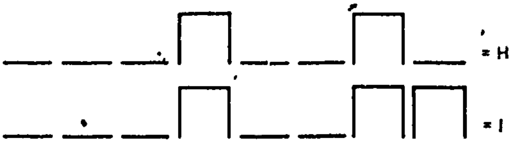


An electronic data processing system, however, must input, store, process, and output information electronically. The symbols humans understand must be represented electronically in order for the computer to understand them. But electronic language is very limited--either an electronic pulse is present, or it is not. Thus, only *two* symbols can be used to represent information in a computer system.

Each piece of information (each letter or number or symbol) is transmitted to the computer in the form of electronic pulses. The pulses are in coded sequences representing the data. For example, the letter "H" might be transmitted in electronic pulses in this pattern:



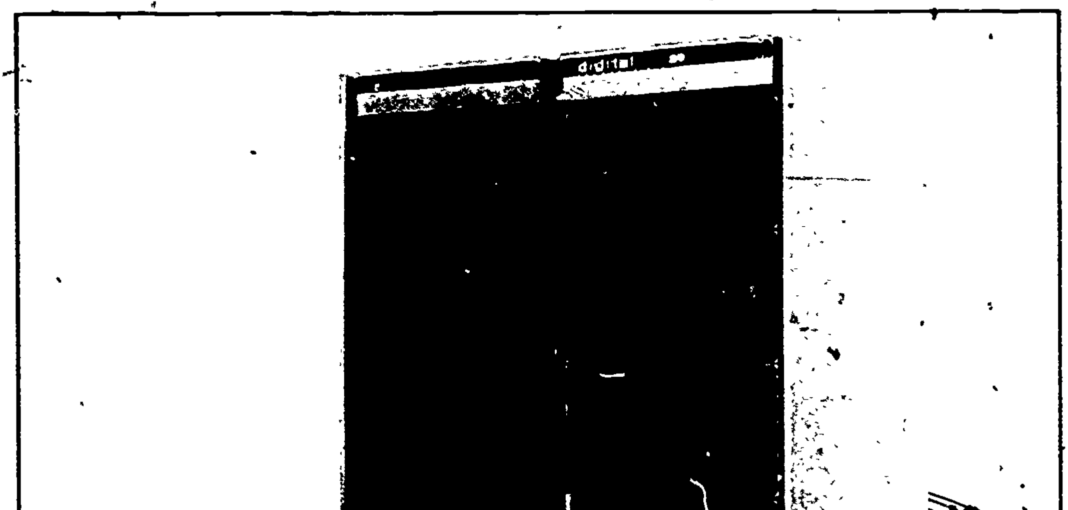
The word "HI" could look like this:



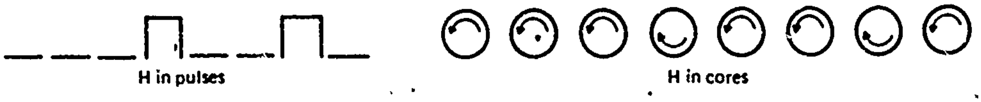
This coded pattern of pulses can be sent over a telephone line from a terminal or read directly into the computer from one of the input devices in the computer center.

Inside the computer's CPU, each coded pattern of pulses is registered in the primary storage unit on very small magnetic "cores" that

Fig. 2-22 Information is input to computers by electronic pulses.



look like tiny doughnuts. Each is about the size of the head of a pin. These little cores can be magnetized in either a clockwise or counterclockwise direction. By being magnetized in one or the other direction, a set of cores can represent a pattern of pulses accurately.



If we represent a pulse with a "1" and a no-pulse with a zero, we could write the word "HI" this way:

| 00010010 | 00010011 |
|---|---|
| H | I |

In fact, we could make up a code, so that every letter of the alphabet and each digit and each punctuation mark cou'd be represented by some unique combination of 1's and 0's. How many different code combinations would we need? Count them. We would need 26 different code combinations for the letters of the alphabet, 10 more for the digits 0–9, and maybe a dozen or so more for punctuation marks and special symbols like $.

This is exactly what has been done! Codes have been developed based on just these two symbols, 0 and 1, for use in communicating with a computer. Of course, humans being what they are, there is no single, standard code that applies to all computer systems or to all input/output mediums like cards or tapes. However, it is important to remember that whatever the code, it is based on combinations of 1's and 0's.*

The codes used by different input devices are usually different. There is one code used for punching cards and another used for punching paper tape. And the codes used to store information in primary storage or on magnetic tape are still different ones. The important thing is that all codes in a computer system can be translated into other codes. We humans understand several codes for the number five: 5, V," We can quickly translate "  " into "5." Likewise, a computer can rapidly translate from punched card code to magnetic tape code.

* If you are interested in codes and want to know more about this two-symbol system, see the Appendix, "Binary Codes."

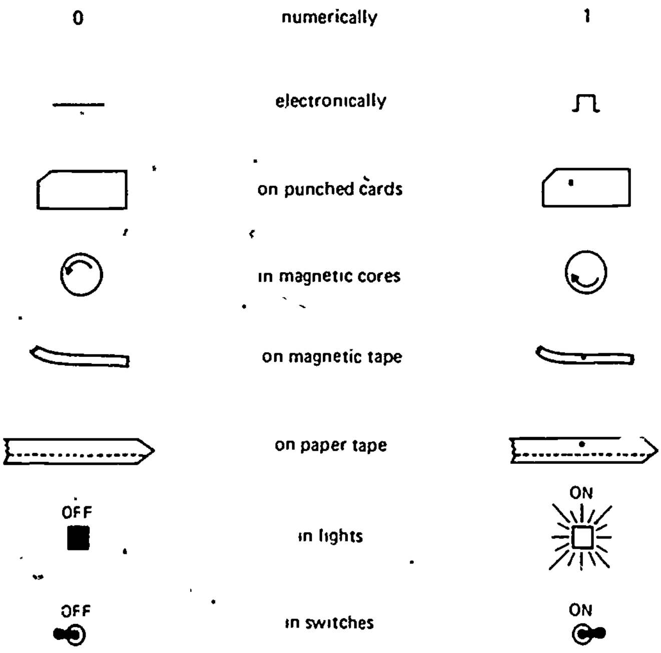


**Fig. 2–23** Ways of representing data in binary code

Whatever the code used, it is a *binary* code [°]; that is, it is a combination of 1's and 0's, or pulses and no-pulses, or holes and no-holes. It is a two-symbol code. Fig. 2-23 shows some ways of representing data in a computer system using only a binary code.

Which ways of representing 1's and 0's in Fig. 2-23 would be used to store data in the primary storage unit within the CPU? Which would be used to display data on a control panel with banks of lights? To transmit data over a telephone line? To transmit data from an electric typewriter through a cable to the computer? To store data in secondary storage? To read input data?

Actually, the same little piece of code, a 1 or 0, might be represented in several ways as it passes through the computer system. For each device (the card reader, magnetic tape unit, primary storage, or output printer), the binary number may be translated to a different code. But the human user of the system is unaware of all this. He or she simply prepares the data and inputs it, then reads the output.

How do programs and data get converted into electronic pulses? That is the job of input devices. Once programs and data are prepared, the cards or tape are fed into an input device which translates the data

[°] "Binary," literally, means "two-state."

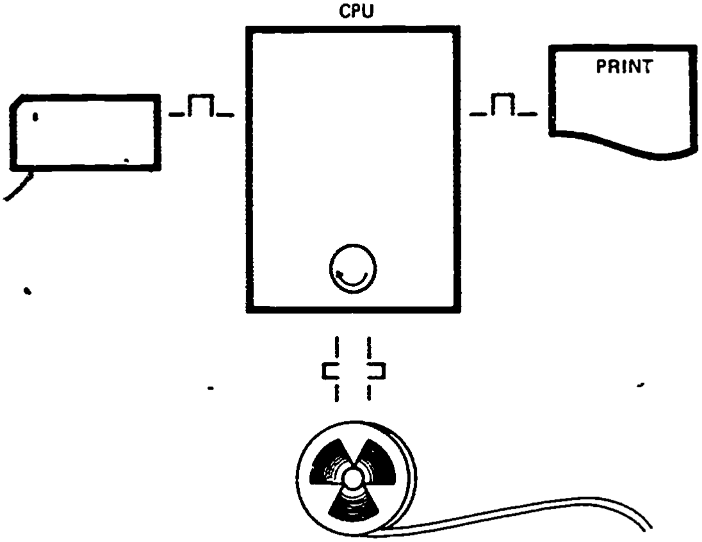


**Fig. 2-24** Binary code can be represented in many different ways in the computer system.

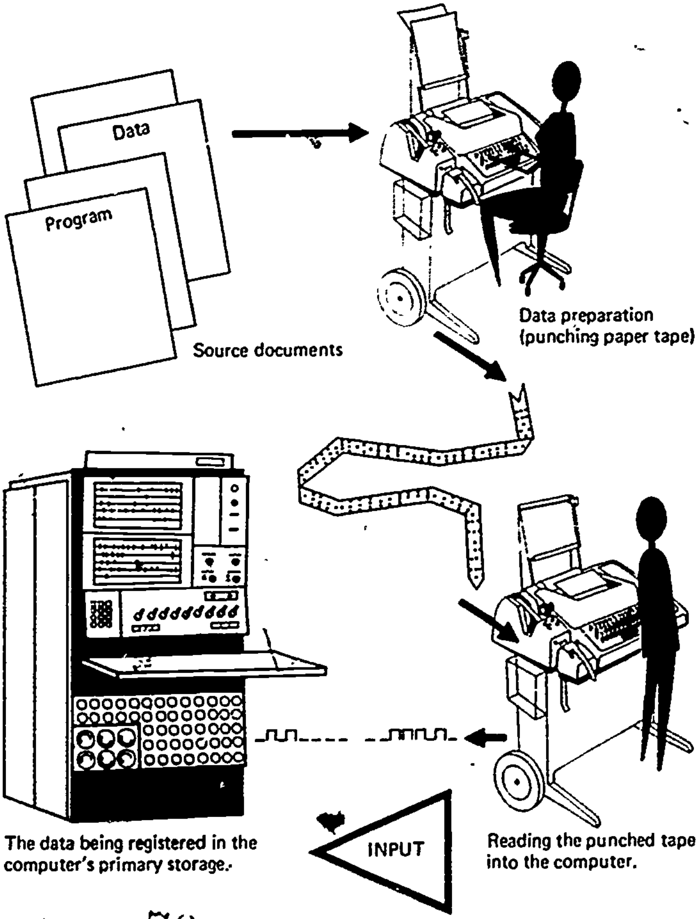

Data
Program
Source documents
Data preparation (punching paper tape)
The data being registered in the computer's primary storage.
INPUT
Reading the punched tape into the computer.

**Fig. 2-25** The flow from raw data to primary storage in the computer

into electrcnic pulses and transmits them to the computer. Now the question is, How has the data been prepared so that it can be "read" and translated by an input device?

The answer is, Data are prepared on the media in the form of spots, marks, or holes arranged in coded patterns. These coded marks are sensed by the input device's "read" mechanism and translated into equivalent electronic pulses. The electronic pulses are transmitted to the computer's main storage. Fig. 2-25 on page 63 shows a graphic picture of this flow with punched paper tape as input.

Currently, the "data preparation" step may not be necessary if a direct-entry input device is used. A person using an electric typewriter terminal, for example, may simply type data at the keyboard. The typed characters will automatically be converted to electronic pulses and transmitted to the computer.

How do the electronic pulses produced by the computer as output get converted into a form humans can read? That is the job of output devices. A printer, for example, was designed exclusively to convert or decode output from electronic pulses to human language.

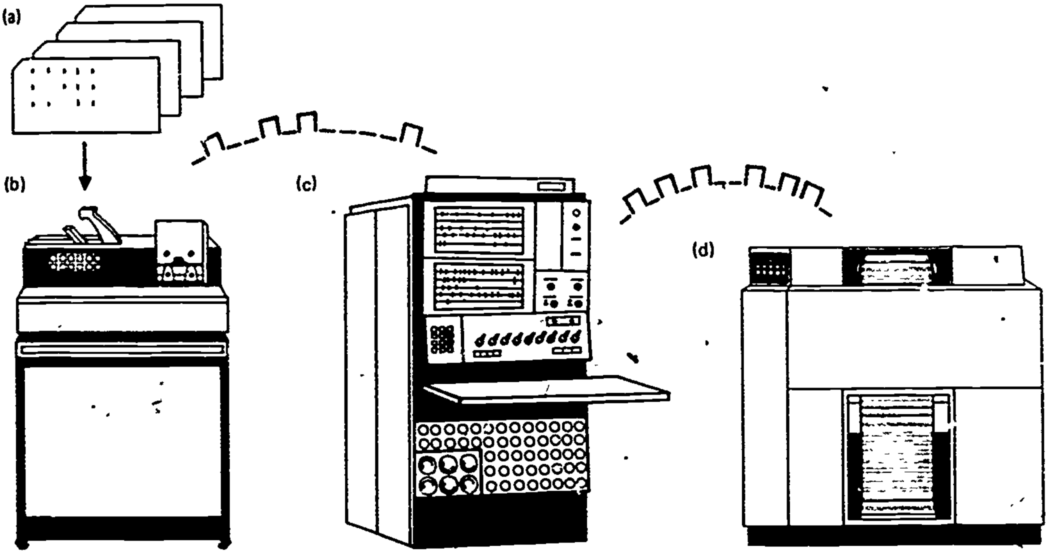
You will learn more about input and output devices in the next two chapters.

Let's see if you have gotten the main ideas about the code used by computers and the different ways it can be translated. Do the following exercises to check your understanding.

## Check your understanding

1. Look at the diagrammed sequence (a) through (d), shown at the top of the next page, and read the descriptions of events a through d. Note for each step (a)-(d) which description fits it.

   a. Holes or spots are read by devices and translated into electronic pulses the computer can interpret.
   b. Information inside the computer is transmitted to output devices in the form of electronic pulses..
   c. Data are prepared by being translated into holes or spots on cards, paper tape, etc.
   d. Data in the form of electronic pulses are registered and stored in the CPU of the computer in the form of tiny magnetized cores.



2. The code used with a computer system is called "binary" code.
    a. How many symbols are there in binary? What are they?
    b. Name five different ways of representing data in binary code.
3. What devices usually are used to convert human data to a code the input device can read and transmit to the computer?
4. In what form is information transmitted to the CPU so the processor can register it?
5. What devices are responsible for converting information into electronic pulses the computer can register?
6. What devices are responsible for converting information from electronic pulses into words or symbols humans can understand?

## DATA PREPARATION

As you are already aware, if you are not using a direct-entry input device such as a typewriter terminal, there is one essential step to be added to the familiar basic steps in data processing.

Data preparation is the act of preparing the *medium* to be read into the computer by an input *device*—for example, punching holes in cards (the medium) to be read into the computer by a card reader (the device). It is important to distinguish between a medium and a device.

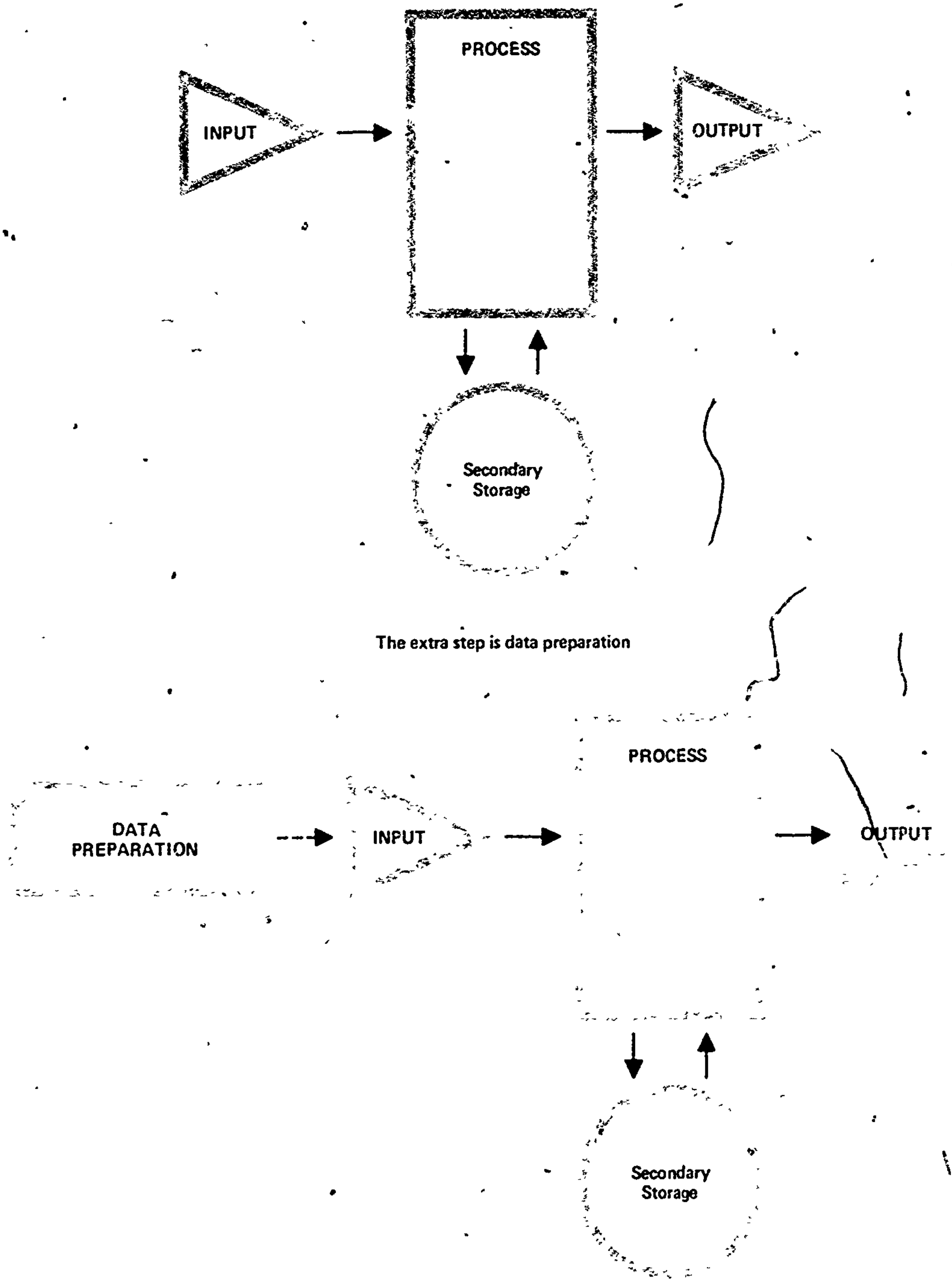


Fig. 2-26 The data preparation step added

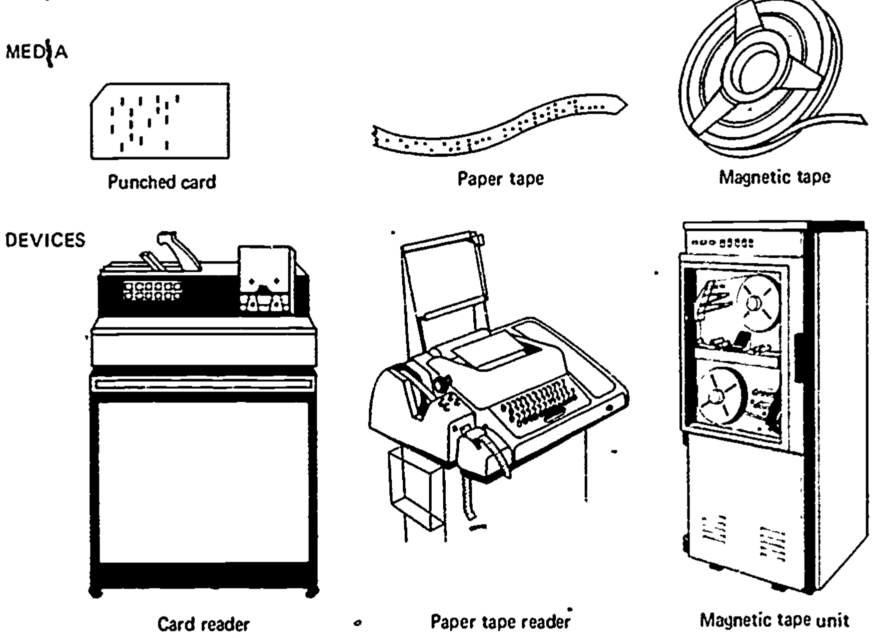
A *medium is a material* (such as punched cards, punched paper tape, and magnetic tape) on which information is represented in coded form. This code can be recorded on the medium or read from a medium, by a *device*.

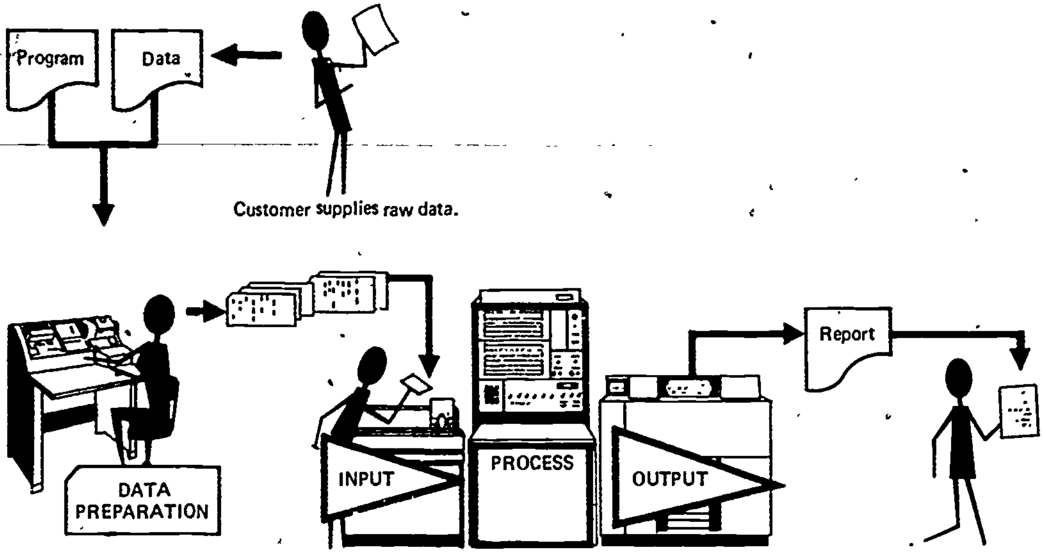
A *device is a piece of equipment* which can record code onto a medium or read code from a medium.

Let's look at the data preparation step as it actually is carried out. We'll take, as an example, data which are prepared on punched cards, because punched cards are probably the most frequently used medium for data input. The flow from the original data through preparation and input to the final output is illustrated in Fig. 2-28.

As you can see from the diagram, the first stages in processing data involve a program, which is written by a programmer, and some data, which are supplied by the customer. The handwritten or typed programs and data are turned over to the person who "prepares" the material for entering into the computer. In our example, it is the keypunch operator who prepares the material on punched cards, using a keypunch ma-

**Fig. 2-27** Some media and the input devices which can read from them





Customer supplies raw data.

Keypunch operator prepares program and data on punched cards.

Computer operator loads program and data into computer.

Customer receives completed report.

**Fig. 2–28** A simple diagram of the flow from data preparation to output

chine. The prepared data (including programs) then go to the computer operator, who runs the job on the computer. If there are no errors in the data or programs entered, the output of the processing can be delivered to the customer. If there are errors in the data or programs, the cards are returned to the programmer so the data or programs can be fixed or "debugged." Then, corrected data and/or programs are prepared and processed. You will learn more about "debugging" in Chapter 5.

While the person who prepares the data may sometimes have to prepare hand-marked media such as mark-sense cards, most often the person works with machine-marked media such as punched cards or paper tape. The machines used to prepare these media are usually run from typewriter-like keyboards on which the data are typed.

A person who punches cards is usually called a "keypunch operator." We will use the more general term "data clerk," however, when referring to the person who prepares data, because we will be discussing how data are prepared on paper tape and other media as well as on cards.

## THE MEDIA OF DATA PREPARATION

### Punched cards

No doubt you have seen examples of the commonly used medium of punched cards. Today many companies use them for billing. Your telephone bill, for example, probably comes with a punched card which is to be returned with the payment. Perhaps your school issues punched cards at the beginning of the term for students to use as class admission cards. It is even common for paychecks to be printed on punched cards.

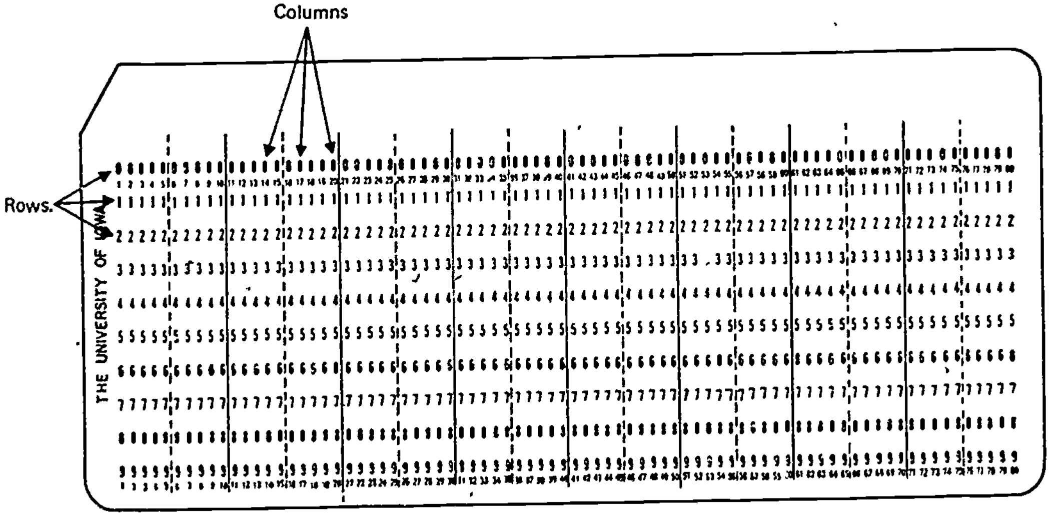
Not only are punched cards used for input, but they are used also as output media. Sometimes the computer is directed to punch output into cards for later use by the system.

The cards used as input media are sometimes called Hollerith cards, after Dr. Herman Hollerith, a handsome gentleman with a handlebar mustache, who built one of the first card-punching machines for use in the 1890 U.S. census. The code of holes used to represent data in punched cards is called Hollerith code.

Here is a blank Hollerith card with a left-corner cut. This cut helps the operator do a quick visual check to make sure all the cards in a deck are facing the same way.

Notice that the card is divided into 80 columns, numbered left to right, 1 through 80. Each column can hold one character of information, so up to 80 characters (letters, numbers, or symbols) can be punched in each card.

Fig. 2–29 Standard Hollerith card



***Numbers—Hollerith code.*** Let's look at some numbers punched in a card. The Hollerith code for numbers is very simple. Each digit, 0–9, is represented by a punch in the corresponding row. Can you see that the number "6501" is punched in the card in Fig. 2-30?

Take a blank card and, using a pencil, darken the numbers corresponding to your telephone number, including the area code.

Fig. 2–30 Card punched with data: 6501

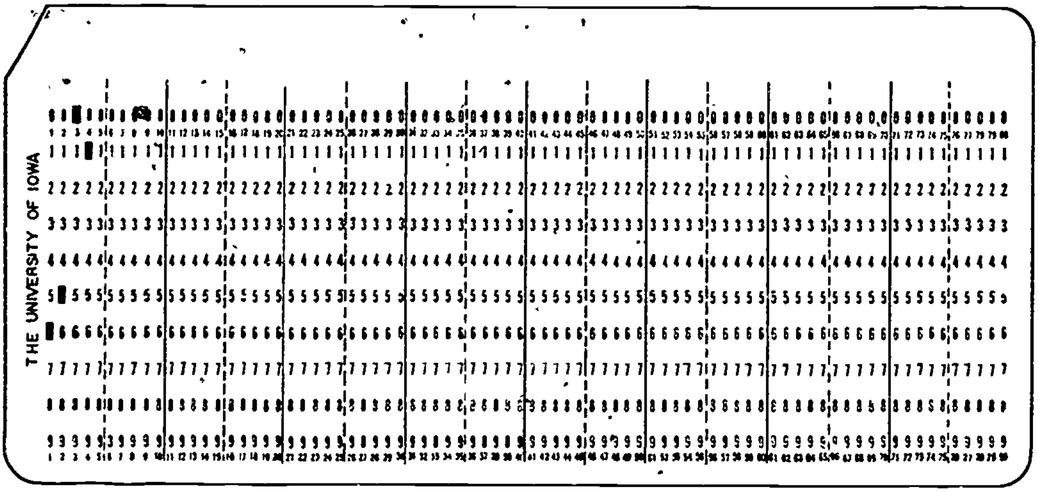
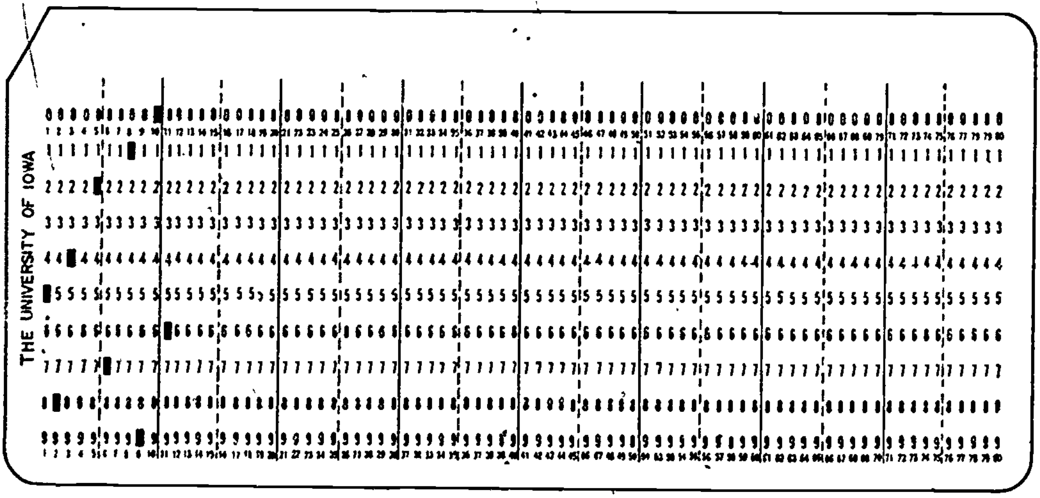


Fig. 2–31 Hollerith card punched with a number



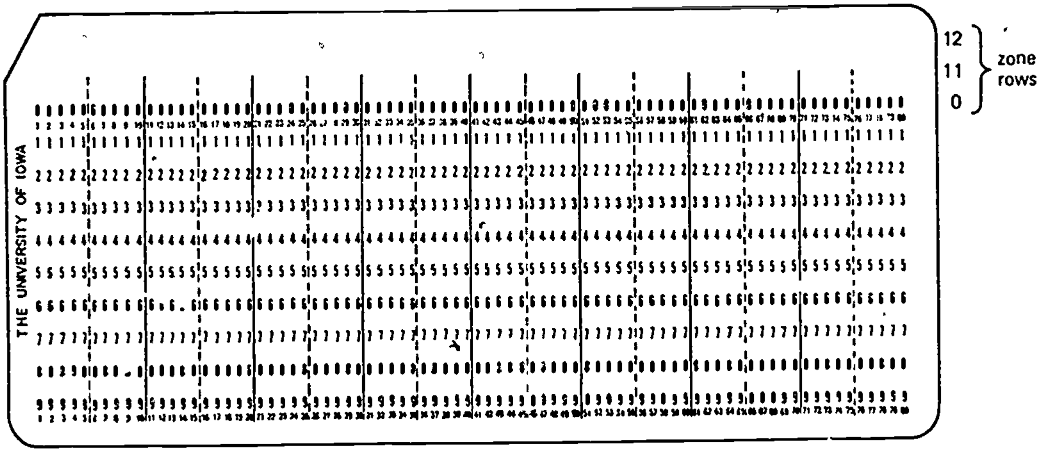


**Fig. 2-32** Hollerith card showing zone rows

Did you use the first ten columns? Did you darken a single digit in each column?

Read and write down the social security number punched in the card in Fig. 2-31.

*Letters—Hollerith code.* To represent a letter of the alphabet in Hollerith code requires *two* punches in a single column: a digit punch together with a punch in one of the top three rows. These top three rows, called "zone rows," are the 12 row (the top row, not labeled on the card), the 11 row (the second row from the top, also not labeled on the card), and the 0 row (which is used when punching either numbers or letters). See these rows in Fig. 2-32.

The first nine letters, A through I, are represented by a zone punch in row 12 together with a digit punch. The letters J through R are represented by a zone punch in row 11 together with a digit punch. The letters S through Z take a zone punch in row 0 plus a digit punch. (A punch in row zero is called a "zone punch" when used in representing a letter, and a "digit punch" when used alone to represent the number zero.) This code is summarized in the card shown in Fig. 2-33. As you study this and the other sample cards, remember that the zero row can be used for either zone (when a letter is being coded) or digits (when a zero is being coded). You can quickly tell how the zero row is being used when it is punched. If there is no other punch in rows 1 to 9 in the column, the zero punch indicates the digit zero. If another punch is there, then the zero punch is a zone punch.

Perhaps you can see more clearly the "two punches in a single column" rule in Fig. 2-34, showing the letter "H" in Hollerith code. Note the zone-punch in row 12, together with the digit punch in row 8.

Fig. 2–33 Hollerith card punched with A-Z

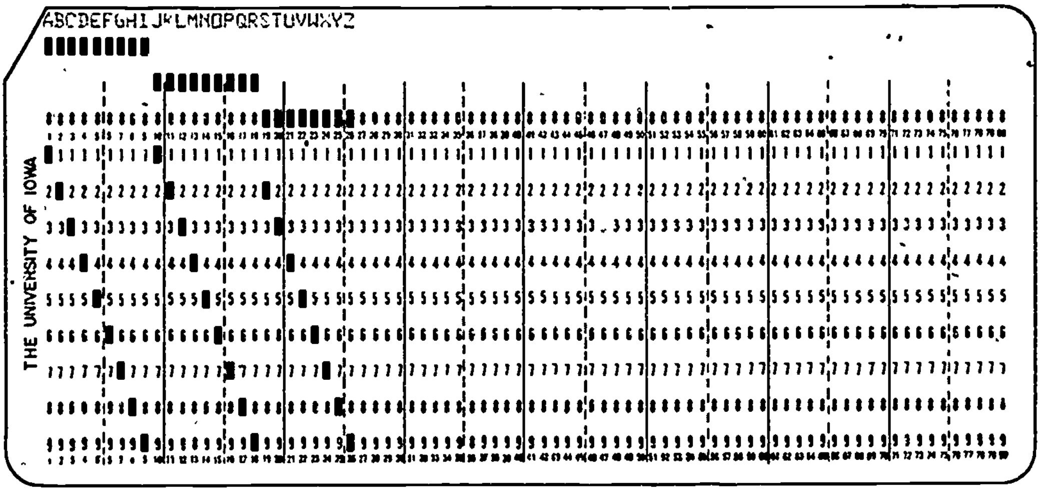
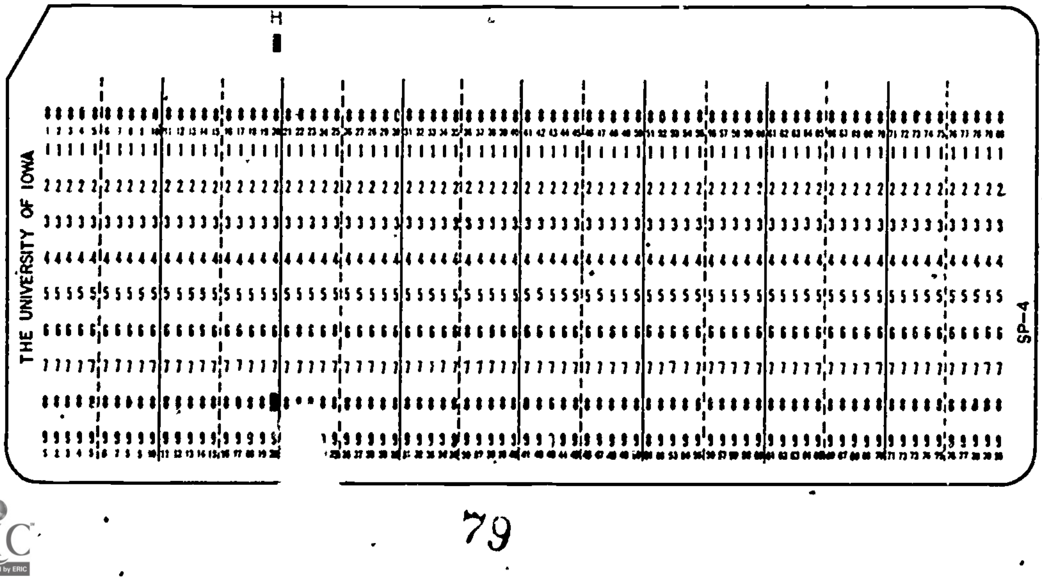


Fig. 2–34 Hollerith card punched with H



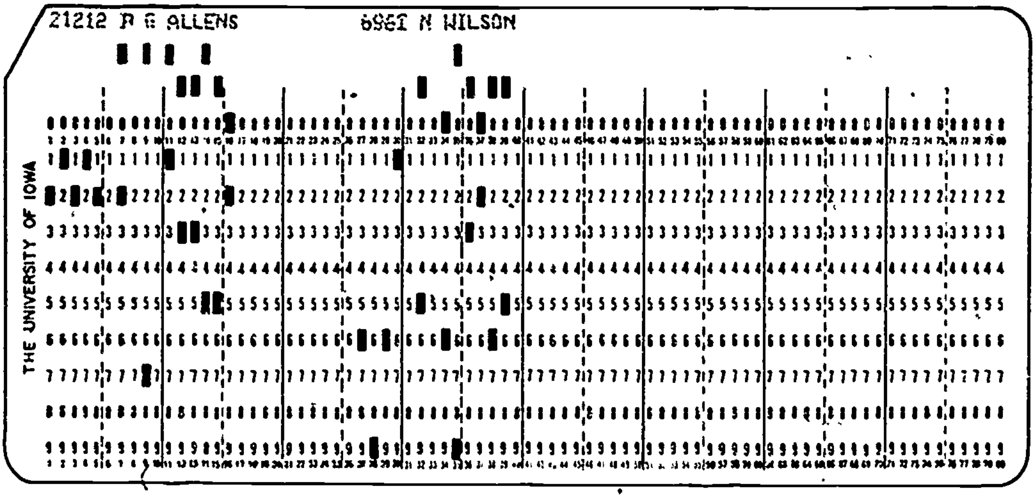


**Fig. 2–35** Hollerith card punched with numbers and letters

Fig. 2-35 shows a card punched with the data

"21212 B G ALLENS        6961 N WILSON"

Can you verify that the data is punched correctly?

Take a blank card and, using a pencil, darken the positions corresponding to the letters of your name. Leave a space between your first name and your last name.

Did you darken two "punches" for each letter?

Did you use a single column for each letter?

Read and write down the data punched in the card in Fig. 2-36. Use the card in Fig. 2-33 as a guide.

*Special characters—Hollerith code.* To punch some punctuation marks and other special symbols, *three* punches may be required in a single column. For some special symbols, only one or two punches are needed. The code for special symbols is shown in the card in Fig. 2-37. Using the guide shown in Fig. 2-37, can you tell which special symbols are encoded on the Hollerith card in Fig. 2-38?

The symbol in column 20 is a "?" and the symbol in column 40 is a "–" (minus sign). The question mark (?) is represented by three punches in a single column: the 0 zone punch, the 7 punch, and the 8 punch. The minus sign (—) takes only a single punch—the 11 zone punch.

Now, you should be able to read any sort of data punched in a card by decoding the Hollerith code. Try it with your next telephone bill!

**Fig. 2–36** Punched Hollerith card

6

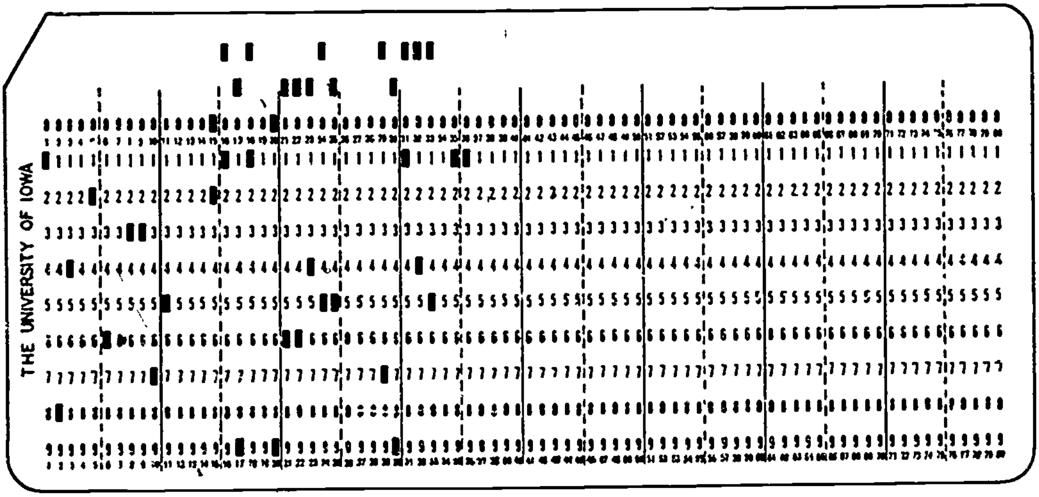
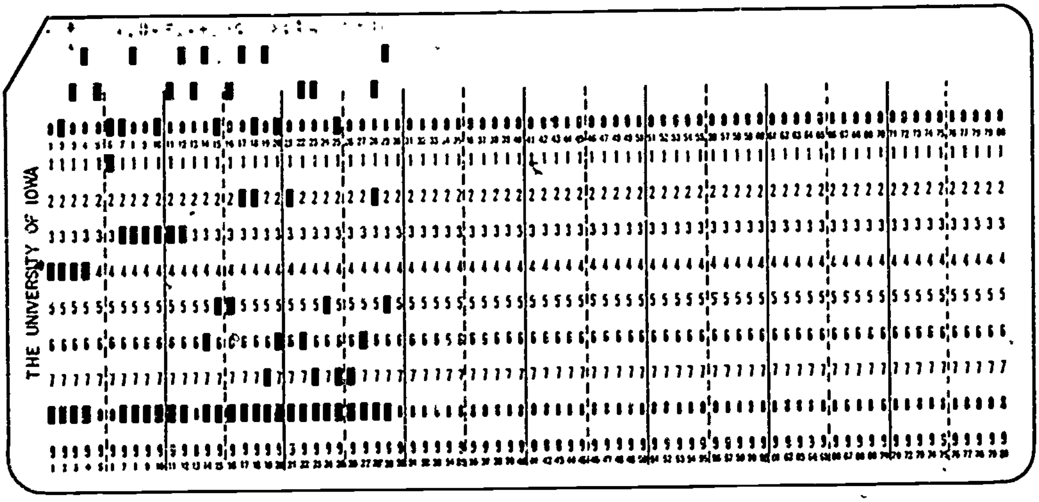


**Fig. 2–37** Hollerith card punched with special symbols



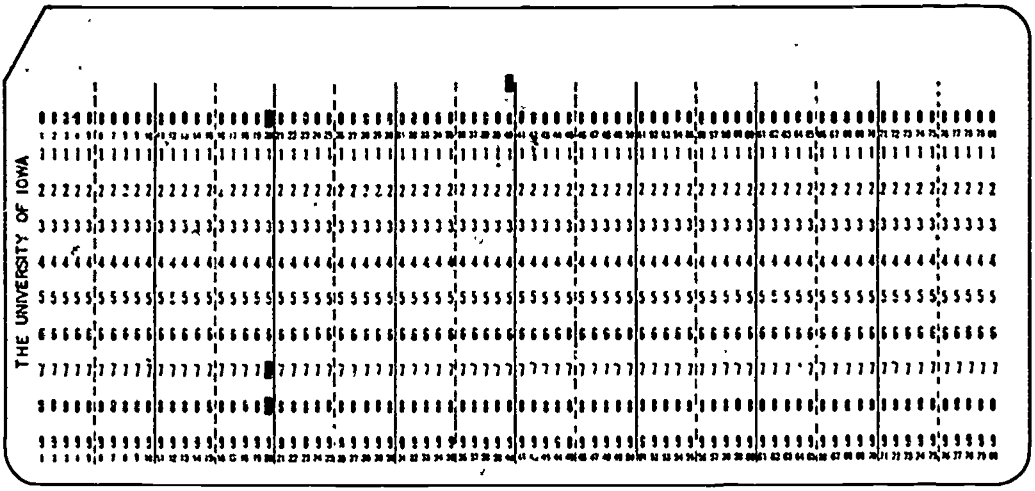
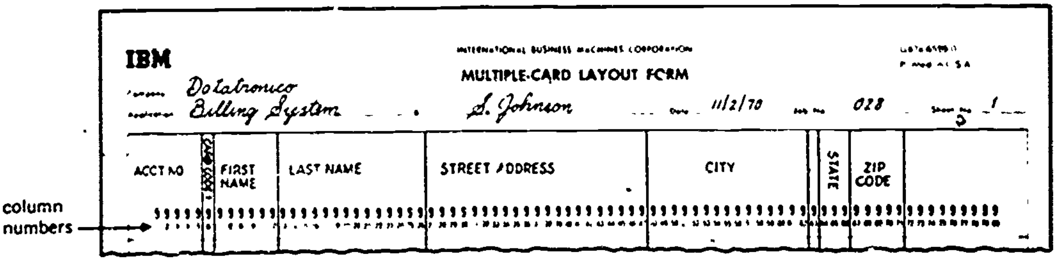


Fig. 2-38 Hollerith card punched with two special symbols


IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

MULTIPLE-CARD LAYOUT FORM

Dotatronico

Billing System

S. Johnson

Date 11/2/70

028

1

| ACCT NO | | FIRST NAME | LAST NAME | STREET ADDRESS | CITY | | STATE | ZIP CODE | |
|---|---|---|---|---|---|---|---|---|---|

column numbers

Fig. 2-39 Sample card layout.

*Fields.* When preparing data, the data clerk always needs clear instructions on how data are to be spaced and arranged. For exampie. programs usually are written on coding forms which show the needed details of arrangement. In the case of data to be prepared the programmer must provide a card iayout and 'or a special instruction sheet for the clerk to follow.

Whether on a coding sheet or card layout, data are always arranged by column number. Look at the sample address card layout for a billing system in Fig. 2-39.

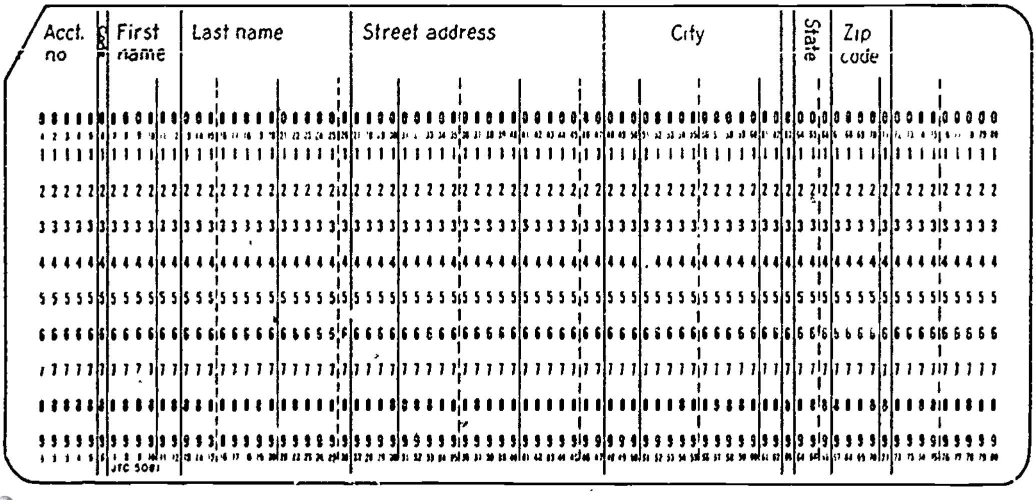
The sections of the card laid out in Fig. 2-39 are

| | |
|---|---|
| Columns 1–5 | Account Number |
| Column 6 | Card Code |
| Columns 7–12 | First Name |
| Columns 17–26 | Last Name |
| Columns 27–47 | Street Address |
| Columns 48–62 | City |
| Columns 64–66 | State |
| Columns 67–71 | Zip Code |

In a card layout such as this, each of the sections is assigned to a particular piece of data—account number, card code, etc. These sections are most often called "fields." The fields shown on the layout card in Fig. 2-39 can be blocked-off (or "defined") on a punched card as shown in Fig. 2-40.

Whenever the programmer has laid out and defined the fields for the preparation of a particular set of data, the data clerk must be certain to prepare each bit of data within the boundaries of the correct field. If a piece of data is entered in the wrong field, or if it is prepared so as to extend into the next field, the computer will either not accept the data at all or it will err in the use of the data. For example, if a street address entered on the card in Fig. 2-40 were to extend from column

**Fig. 2–40** Example of fields marked off on a punched card



27 to 55 (eight columns into the city field), the computer would register the last seven letters of the street address as the first seven letters of the city name.

Check your understanding now.

# Check your understanding

1. Data preparation is *usually* not necessary when one is using (as an input device)

   a. a punched card reader.
   b. a magnetic disk.
   c. a magnetic tape unit.
   d. a typewriter terminal.
   e. a paper tape reader.

2. Identify each of the below-named items as either a "medium" or a "device."

   a. Punched card
   b. Magnetic tape unit
   c. Paper tape punch
   d. Magnetic tape
   e. Card reader
   f. Paper tape

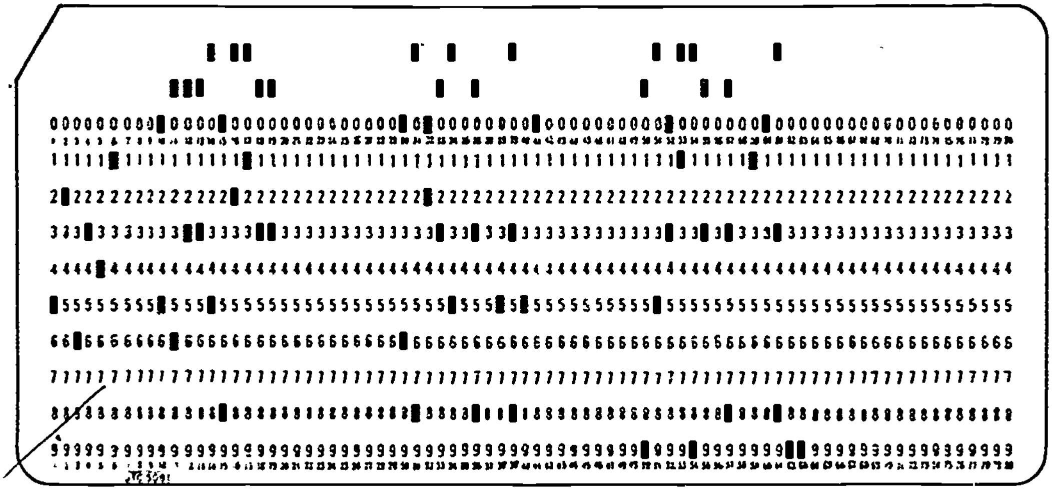
3. A "keypunch operator"

   a. operates the punched card reader.
   b. operates the keypunch machine.
   c. is not a data clerk.
   d. punches holes in paper tape.

4. Answer the following questions about punched cards.

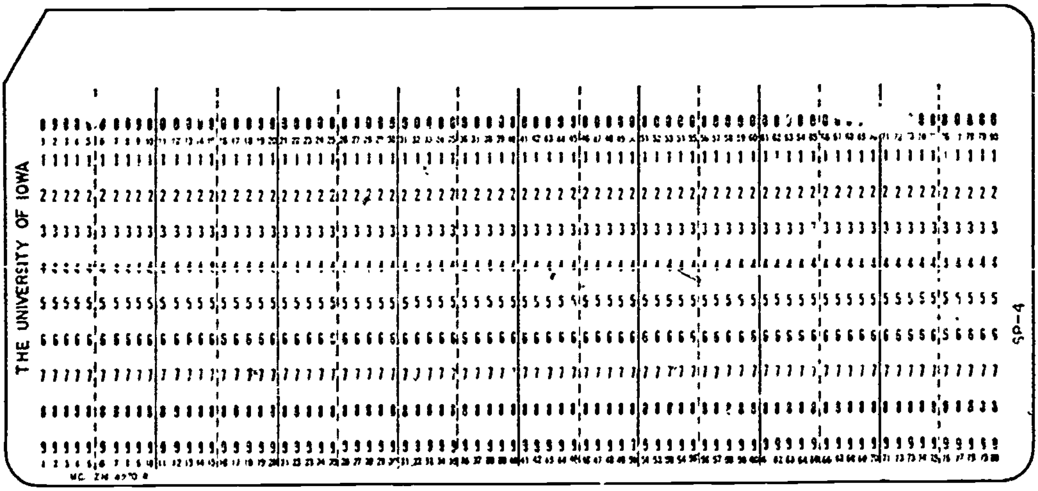
   a. How many columns are there in a typical Hollerith card?
   b. What is the maximum number of characters that can be punched in a single typical card?
   c. How many rows are there in a card?
   d. How are the rows labeled?
   e. Which rows are designated as "zone rows"?
   f. Why is it necessary to use two holes to represent letters of the alphabet?
   g. How many holes per column are needed to represent special characters?

**5.** Read and write down the data punched in the card below.



**6.** With a pencil, darken "holes" in a blank card to represent the following data:

TROUT FISHING IN AMERICA PAPERBACK $2.95



**7.** When preparing data using a keypunch machine, the data clerk

a. need not be concerned about the arrangement of data.

b. rarely needs coding sheets, layouts, or other directions to follow.
c. must be careful to punch data within the correct fields on the cards.
d. must start data in the correct column, but need not be concerned about what column the data ends on.

8. Which of the following is true of "fields" on cards?

a. They are always located by column numbers on a card.
b. Each field has definite boundaries.
c. They are the sections of a card defined by the programmer to contain particular pieces of data.
d. All of the above statements are true of "fields" on cards.

## Optical scan sheets

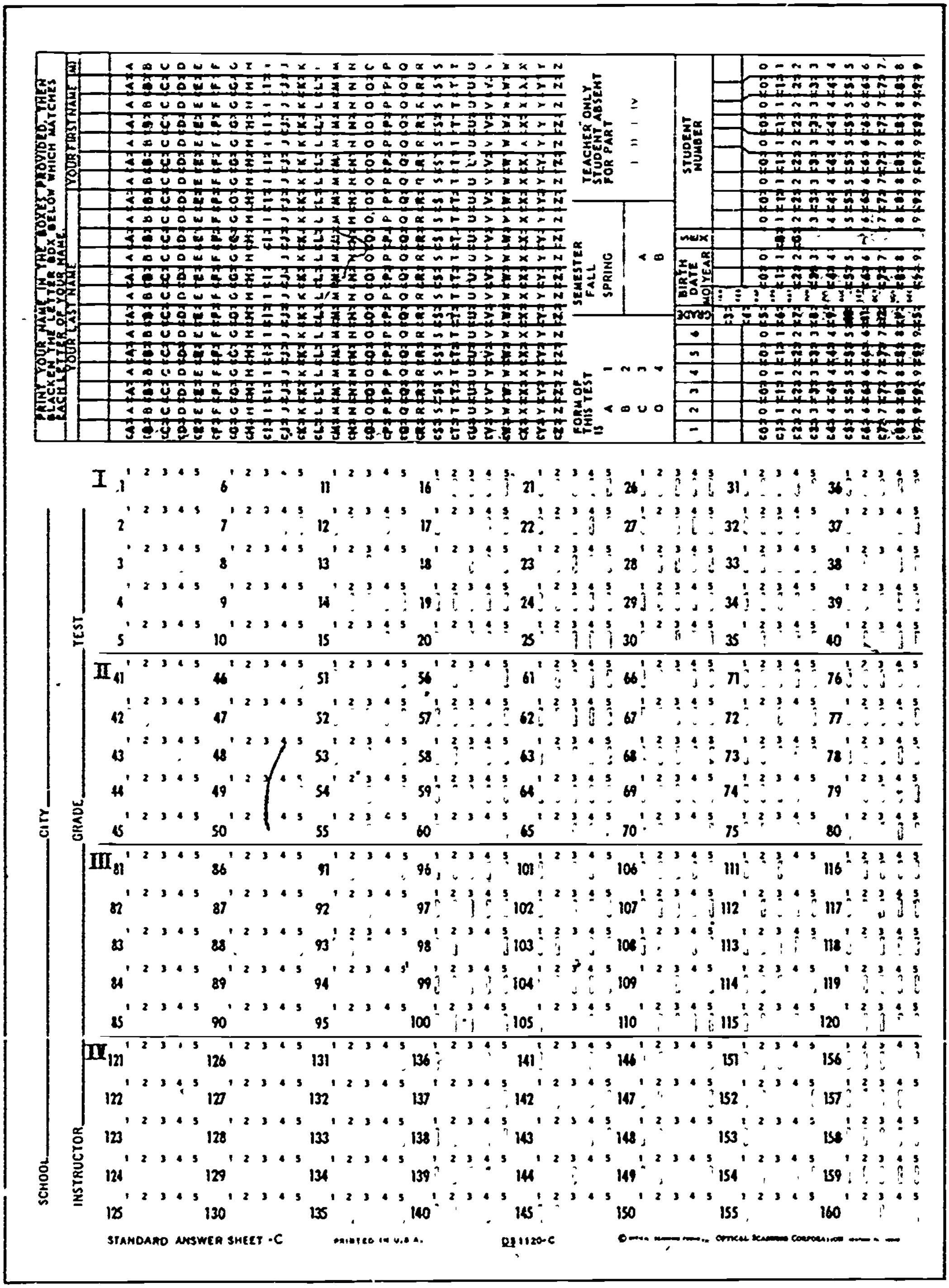
If you have ever taken standardized "achievement" tests, you have marked your answers on a sheet which can be used as an input medium for computerized grading. These sheets, called optical scan sheets, are usually 8½ inches by 11 inches and printed with little boxes. The pencil marks you place inside the little boxes can be read by an input device, just as can holes in cards.

In the past, these scan sheets were "read" electronically, so that a special pencil had to be used for marking. The special pencil had a high graphite content since graphite is a good conductor of electricity. Now, however, these sheets can be marked with any kind of pencil, typewriter, or even punched with holes, since the newer readers simply sense the difference in the reflection of light from the surface of the sheet.

Thus, the data preparation step is eliminated. The "source document" (the scan sheet) is a record that can easily be read and understood by people as well as by computers.

## Punched paper tape

Punched cards are the most commonly used media in computer centers. But punched paper tape has an important place as an input medium as well, particularly outside of the computer center. You will very often find paper tape used as the input medium where data is



PRINT YOUR NAME IN THE BOXES PROVIDED, THEN BLACKEN THE LETTER BOX BELOW WHICH MATCHES EACH LETTER OF YOUR NAME.

YOUR LAST NAME

YOUR FIRST NAME

(M)

FORM OF THIS TEST IS A B C D

SEMESTER FALL SPRING A B

TEACHER ONLY STUDENT ABSENT FOR PART I II III IV

GRADE

BIRTH DATE MO YEAR

SEX

STUDENT NUMBER

SCHOOL

CITY

INSTRUCTOR

GRADE

TEST

I II III IV

STANDARD ANSWER SHEET -C

PRINTED IN U.S.A.

DS 1120-C

© [illegible] OPTICAL SCANNING CORPORATION [illegible]

Fig. 2–41 Test answer sheet

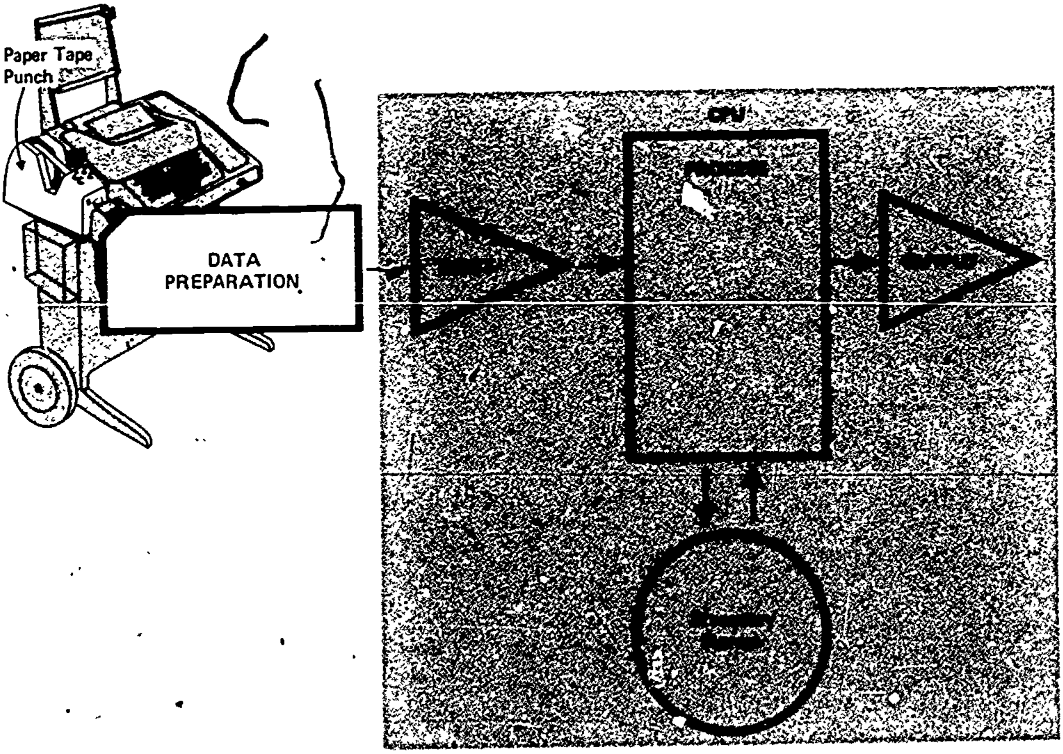
prepared for entry into a computer from a remote, time-shared terminal —particularly if that terminal is a teletypewriter.

The teletypewriter (also called "Teletype" or simply "Tty") is the device most frequently used to punch paper tape. It is actually a dual-purpose device, since it can be used to *punch* the tape and can then be used to *read* the punched tape into the computer.

We'll take a closer look at the teletypewriter later. For now, let's look at the medium—punched paper tape. Fig. 2-44 shows an example. The drive sprocket holes, which run the length of the tape, are not a part of the code punched on the tape. They are used to direct the paper tape through the tape reader. The larger holes, punched in rows across the tape, are the code holes. Each row of holes across the tape represents one character.

A paper tape can be punched automatically using a teletypewriter keyboard and the attached paper tape punch. The machine and the punch are both switched on. Whenever a key is pressed down on the

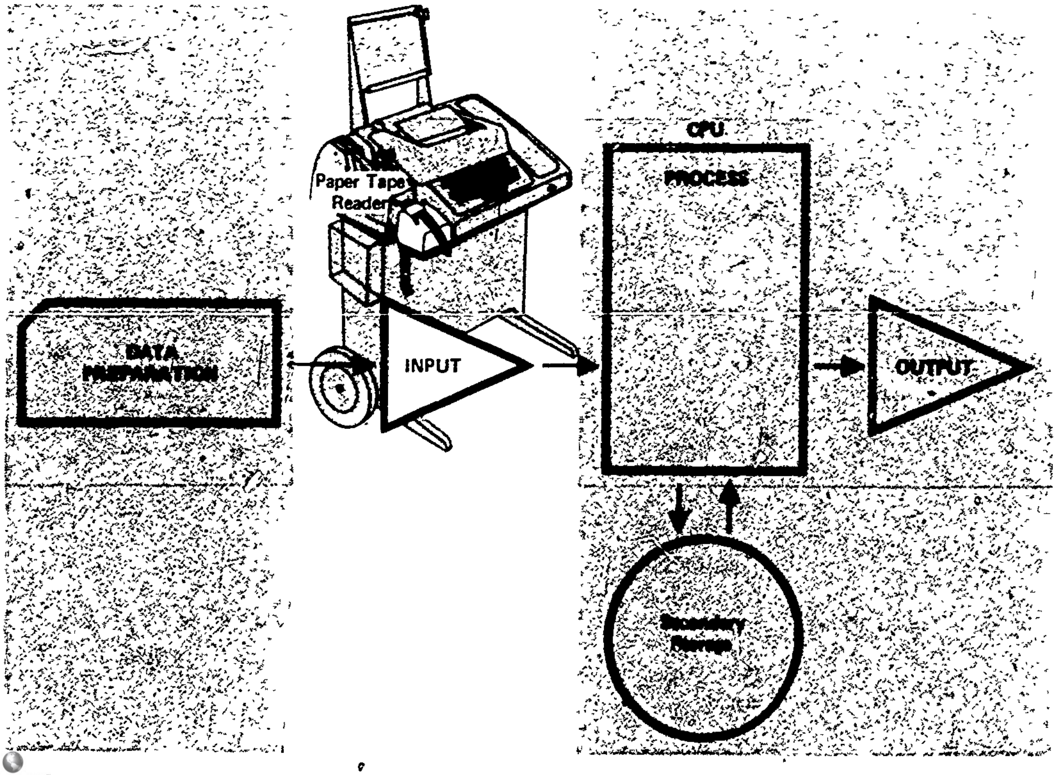
**Fig. 2-42** The paper tape punch device on a teletypewriter



keyboard, it is typed on the teletypewriter's paper. The holes representing that character are also punched onto the paper tape by the punch. The tape then automatically moves forward one position to be ready for punching of the next code.

If you look carefully at the sample tape in Fig. 2-44, you will see that there is room for eight holes to be punched across the tape. However, the eighth hole-position, whether punched or not, is not a part of the code. The eighth position is called a "parity check," and is used to doublecheck the accuracy of the data. Teletypewriters are either even parity or odd parity machines. If the teletypewriter is an even parity machine, for instance, a hole is punched in this eighth hole-position of a row whenever the code for the character represented on that row is composed of an even number of punched holes. An operator preparing data on, or receiving data from, this even parity machine would know that it had been punched or transmitted inaccurately if a punched hole appeared in the eighth position when an odd number of holes appeared in the same row.

Fig. 2-43 The paper tape reader on a teletypewriter



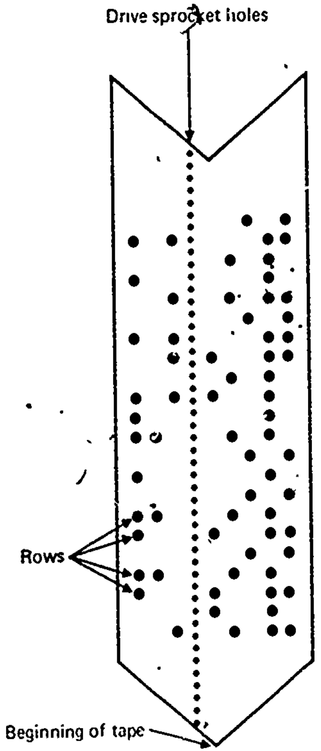


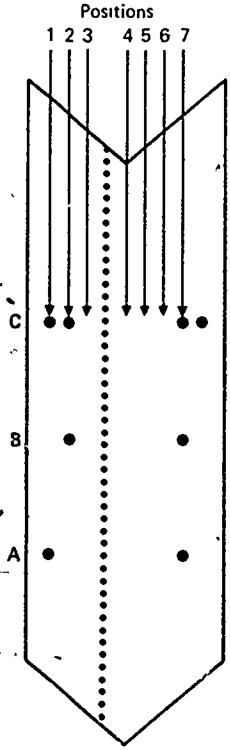
Fig. 2-44 Example of a punched tape with black spots showing punched holes

The code most often used on paper tape is called the ASCII code, which stands for American Standard Code for Information Interchange. ASCII is, of course, a seven-position code. That is, there are seven possible positions for holes. Each character is represented by a unique pattern of holes and non-holes in these seven positions. Fig. 2-45 shows the letters "A," "B," and "C" in these patterns of code holes. The data clerk doesn't need to memorize ASCII code; the paper tape punch automatically punches the correct seven-place code on the tape each time a key is pressed on the keybcard. Sometimes, however, clerks need to read (or decode) a portion of tape, and for this there are tables of ASCII code to refer to. Such a table is included in the Appendix, "Binary Codes." Using this table, verify that the holes punched on the tape shown in Fig. 2-46 are ASCII code for "431-42-9522 JO GIBBS." (Remember that the beginning of the tape is at the bottom marked by the arrow-like point.

The device most commonly used for preparing data on paper tape is the paper tape punch connected to a teletypewriter. Although there are many different kinds of teletypewriters, they all look something like electric typewriters. Many are equipped with both a paper tape punch and a paper tape reader. For our purposes, the teletypewriter must have a paper tape punch and reader attached to it, if we are to use it for data preparation. Such a teletypewriter is shown in Fig. 2-47.

The keyboard of the teletypewriter (or Tty) is arranged very much like that of a conventional typewriter, only some of the special keys at the right and left are used for different operations. Another typing difference is that most teletypewriters (Tty's) type only in capital letters; lower-case letters are not used. There are also some special keys on Tty's which will be explained a little later in this section when you start working at an actual Tty. Right now you need to note that, basically, typing on a Tty is like typing on a conventional typewriter. Everything you type on the Tty keyboard is automatically

**Fig. 2–45** Paper tape showing the seven-code hole-positions for letters A, B, and C



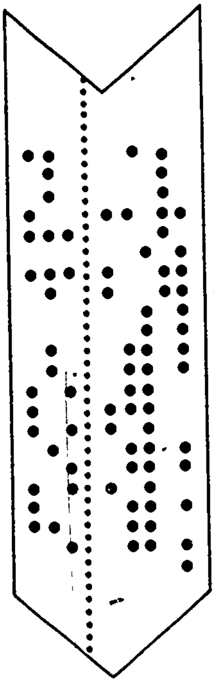


Fig. 2-46 Paper tape punched with data

typed out on a paper above the keyboard. On the Tty this paper is fed in from a continuous roll of paper, whereas on a regular typewriter one sheet of paper is fed in at a time.

Data can be prepared on paper from the Tty's keyboard by switching on the paper tape punch attached to it. Look again at the teletypewriter (Tty) pictured in Fig. 2-47 and locate the paper tape punch on the left side of the machine. As you can see in the illustration, paper tape is fed into the punch from a blank roll and the punched tape is fed out of the front of the punch in a continuous strip.

To punch a paper tape on the Tty, the clerk first switches on the Tty and then the paper tape punch. He or she then proceeds to type the data (accurately) on the Tty keyboard. With the paper tape punch "on" the data are automatically encoded on the paper tape and fed out of the punch. Once a complete set of data has been punched on the tape, the clerk can easily tear the tape off from the punch and switch the punch off.

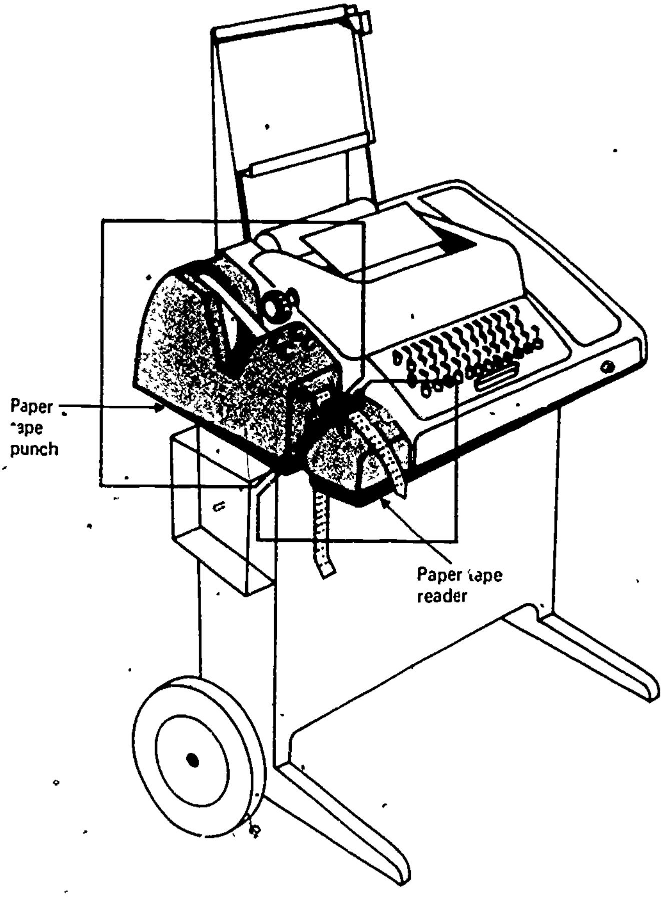


Fig. 2–47 Teletypewriter with paper tape punch and paper tape reader

When a tape has been prepared, it can be checked, using the paper tape reader located just in front of the tape punch as is shown in Fig. 2-47. When a tape is run through this reader, the data on it will be typed automatically onto the Tty paper and this typed version can be checked for errors.

Check your understanding of this material by answering the following check questions.

## Check your understanding

1. When using optical scan sheets (such as a test answer sheet) as the input medium, one step in the data processing routine is eliminated. It is

a. data preparation.
b. input.
c. processing.
d. output.

2. Optical scan sheets are normally encoded with

a. special pencils with high graphite content.
b. electronic markers.
c. regular pencils.
d. special pens with high graphite content.

3. How many positions are used by the ASCII code?

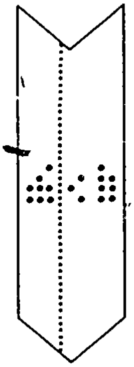
4. There is a row of smaller holes running the length of the punched paper tape; these holes are

a. the sixth hole of the ASCII code.
b. the fourth hole of the ASCII code.
c. feed or guide holes for the tape reader.
d. none of the above.

5. Sometimes a hole is punched at the far right side of the row on the paper tape; this hole is

a. the first hole of the ASCII code.
b. the eighth hole of the ASCII code.
c. feed holes for the tape reader.
d. none of the above.

6. Using the ASCII code table in the Appendix, decode this tape:



What does the tape say?

7. What device is normally used to prepare punched paper tape?

## Magnetic media: tape, disk, and drum

*Magnetic tape.* One of the most widely-used mediums for high-speed input and output is magnetic tape. Besides being considerably faster than pinched cards or tape, magnetic tape can carry much more data in much less space. Compare for example, the 10 characters per inch carried on paper tape with the capacity for up to 1600 characters per inch on magnetic tape (usually half an inch wide). Likewise, while paper tape is read through a teletypewriter at 10 characters per second, data on magnetic tape can be input at the rate of 150,000 characters per second. The content of 10 punched cards (80 c lumns each) can be stored on one inch of magnetic tape and input 50 times faster from tape than from cards.

Another advantage in using magnetic tape is that the tape may be erased and reused over and over again. This makes it an economical medium compared to one-time-only punched cards or paper tape.

Magnetic tape is very similar to the kind of tape used in home tape recorders. Bits of information are recorded as magnetic spots on the tape and can be "read" back to the computer at a later time. These spots, which are invisible to the eye, are recorded in seven "tracks" or "channels" across the width of the tape. (Many types of magnetic tape have nine tracks.) Fig. 2-48 shows a magnified piece of magnetic tape, encoded with the data "ADDRESS FILE."

As you may have figured out, nine bits across the width of the tape represent one coded letter or number—one character—of information. The presence of a magnetic spot represents a binary one and no spots represents a zero.

Although this code may appear to be similar to the ASCII code used on paper tape, it is an entirely different code. It is called EBCDIC

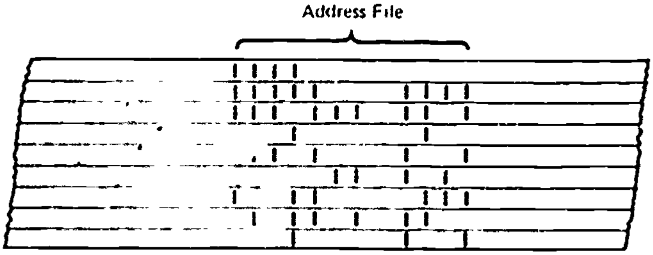


Fig. 2-48 Enlarged magnetic tape encoded with the data "ADDRESS FILE"

(Extended Binary Coded Decimal Interchange Code). It is, however, based on the two-symbol binary system as are all codes used in the computer system. What is the punched card or paper tape equivalent of a "spot" on magnetic tape?

Magnetic tape is wound on "reels." One reel of tape is usually 2400 to 3600 feet long. If one type of magnetic tape can carry 1600 characters (1600 seven-bit patterns) per inch, about how many separate letters and numbers could be carried on a 2400-foot roll of tape?

***Random access devices.*** Although there are many advantages to using magnetic tape as an input medium, there is one drawback: data must be recorded and read *serially*. That is, to get a particular set of data or "record," all previous records preceding the desired record on the tape must first be read in order to get it.

This problem of serial access led to the development of *random access* devices, on which any record can be read without regard to its order in the set of records. Random accessing is like placing a phonograph needle in the middle of a long-play album to play *just* the third song. The first two songs, therefore, don't have to be played before the third is finally reached. Instead of phonograph needles, random access devices have what are called "read-write heads." As the term implies, these heads both record and read data on the particular random access device used. Random access is often referred to as directly addressable or direct access data.

Two magnetic mediums which allow random access, and thus can be faster as an input medium, are *magnetic disk* and *magnetic drum*.

***Magnetic d'*** . The magnetic disk is similar to the old-fashioned jukebox record. However, if you look carefully at the disk illustrated in Fig. 2-49, you will notice that the tracks on the disk actually form separate circles or tracks rather than spiralling in a continuous line toward the middle, as grooves on records do. One disk, in fact, can have hundreds of tracks on each side. Data are recorded on these tracks in code by read-write heads in the form of magnetic spots. The coded spots are almost microscopically small.

To make them even more useful and easy to handle, disks are stored and worked with in detachable "packs." Several disks are stacked together in one pack with a fixed distance of ¼ inch or so between each disk. A disk pack will normally contain between six or more separate disks, and can hold up to nearly eight million characters (or as few as a quarter million characters), depending on the number of tracks on each disk and the number of disks in the pack. A typical disk pack might have six disks (ten recording surfaces since top and bottom are not used for storage of data) with 7,250,000 characters of information.

Since the surface of each disk has a magnetic coating onto which the coded spots are written, you can understand that care is needed

Fig. 2-49 Magnetic disk, showing recording tracks

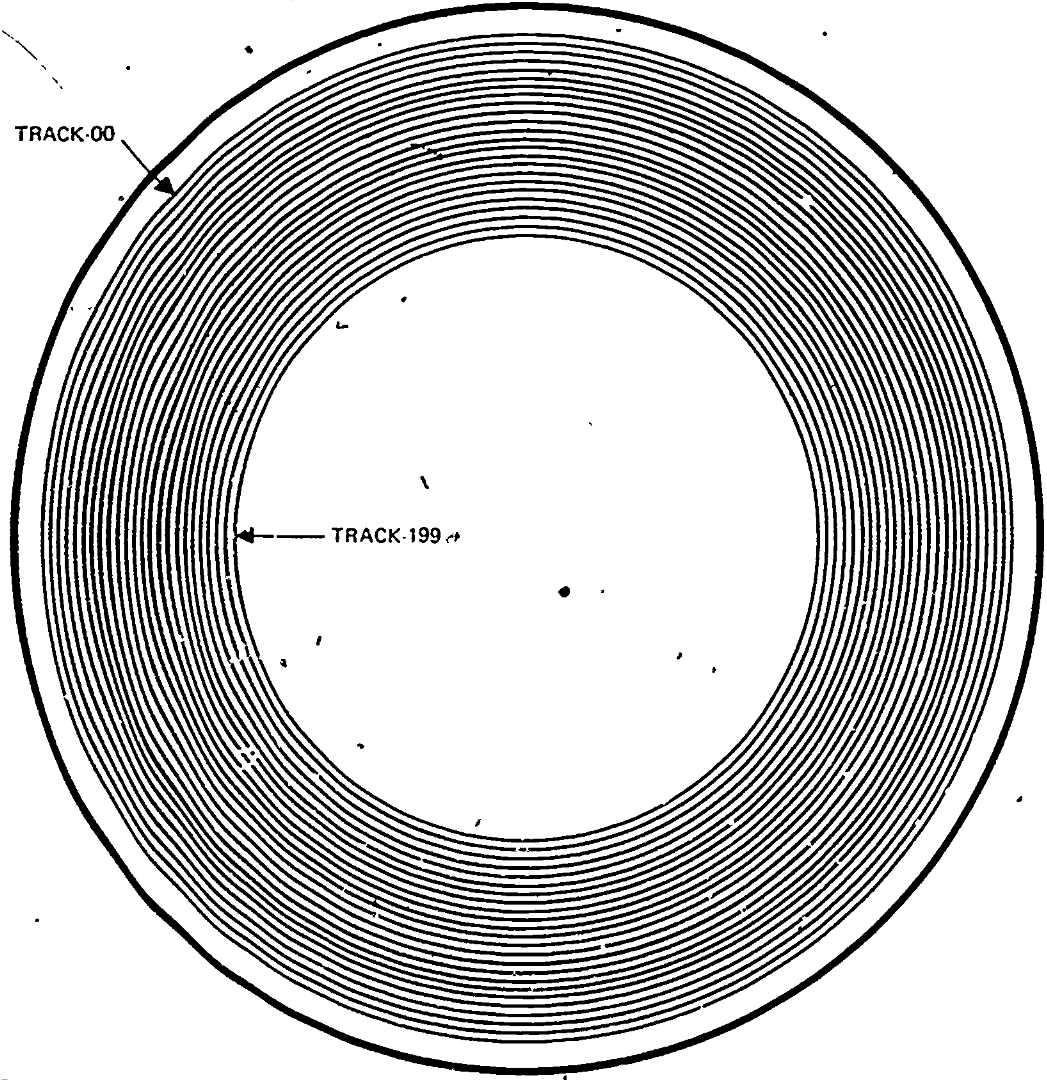


Fig. 2-50 A typical disk pack

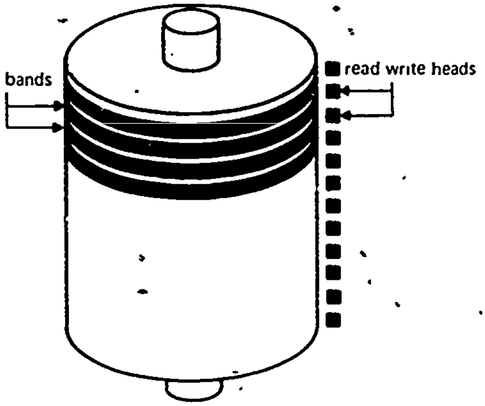


Fig. 2-51 Magnetic drum

to protect the surfaces from damage. For this reason, packs are kept in plastic covers when not in use. An added protection is given the disks in disk packs by leaving the two exposed disk surfaces (the top surface of the top disk and the bottom surface of the bottom disk) blank.

*Magnetic drum.* A magnetic drum is a metal cylinder coated with magnetic material. While it may be of almost any size, the typical magnetic drum is about the size of three typical disk packs stacked one on top of the other. Data are recorded as magnetized spots in circular tracks or bands around the surface of the drum, as shown in Fig. 2-51.

Before going on, check your understanding of magnetic media by answering the following questions.

# Check your understanding

1. Data on magnetic tape can be input at the rate of 150,000 characters per second. Which of the following media can be used for faster input?
   a. Magnetic disk
   b. Paper tape
   c. Punched card
   d. Magnetic drum

2. One big disadvantage of magnetic tape as an input medium is
   a. the high cost.
   b. lack of durability.
   c. random access.
   d. data are recorded and read serially.

3. Which pair of input devices below is an example of random access?
   a. Punched card reader and paper tape reader
   b. Magnetic tape unit and magnetic disk unit
   c. Magnetic disk unit and magnetic drum unit
   d. Magnetic drum unit and magnetic tape unit

4. A disk pack usually contains
   a. a pack of disks.
   b. 6 or more separate disks on one spindle.
   c. probably a million characters.
   d. 5 to 100 tracks

5. On a magnetic disk, data are stored
   a. in a continuous spiral.
   b. as magnetized spots on strips of tape.
   c. in magnetized stacks.
   d. as magnetic spots along circular "tracks."

6. On a magnetic drum, data are stored
   a. in circles on flat discs.
   b. in vertical strips around a circular device.
   c. on circular bands around a cylinder.
   d. in a random pattern on the drum surface.

7. Describe the difference between random access and serial access.

## Direct entry data preparation devices

At the time when computers were first manufactured for wide use, keypunch machines were already long-familiar devices. Even before computers, people were processing data by sorting and rearranging punched cards, and punched cards had been in use since Herman Hollerith's 1890 invention. It was easy, then, to convert punched cards to serve as an input medium for the computer.

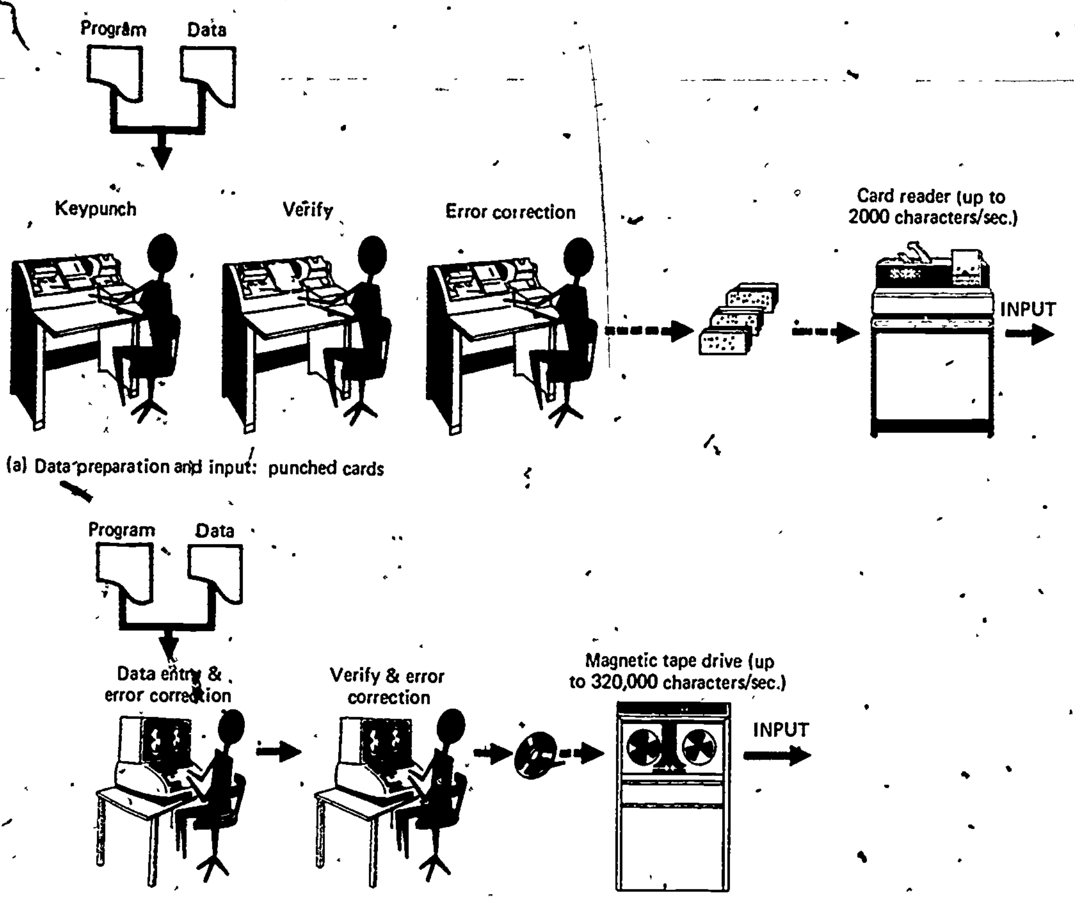
There is much to be said in favor of this medium for computer input. The punched card is a highly accurate input medium. Errors can be corrected easily. The data on the card are visible and can be read by human beings. The punched card provides a single, convenient record which is particularly useful as a bill or a paycheck. In addition, the keypunch is an economical and sturdy device for data preparation.

As CPU processing speeds have increased, however, and the amount of data to be processed has grown to great volumes, the physical handling required in preparing punched cards creates a slowdown or "bottleneck" in the input-process-output cycle. The data preparation requires keypunching, verifying (by punching all the data over again), and correction of errors (by punching new cards), plus physical handling of stacks of cards. If a card is accidentally mutilated, bent, stapled or spindled, it stops the punched card reader and thus halts the entire input operation. Furthermore, the punched card is "slow." Data input from cards can be read at speeds of up to only 2000 characters per second, while magnetic tape, for instance, can be read at a rate of up to 320,000 characters per second.

Thus, twenty years after the development of the modern computer, more efficient methods of preparing data for entry have been developed in the form of direct data-entry devices.

Direct data-entry devices eliminate the keypunch by recording data *directly* on magnetic tape or magnetic disk. Such devices speed up data preparation by allowing a record to be corrected as it is verified, saving a step for the operator. A direct-entry device, because it operates electronically, is faster than a mechanical keypunch machine. Input data recorded on magnetic tape, then, can be input several hundred times faster than punched cards, with no chance of a damaged card stopping the input operation.

Let's take a closer look at the direct data-entry device called a "key-to-tape" device.


Program
Data
Keypunch
Verify
Error correction
Card reader (up to 2000 characters/sec.)
INPUT
Program
Data
Data entry & error correction
Verify & error correction
Magnetic tape drive (up to 320,000 characters/sec.)
INPUT

(a) Data preparation and input: punched cards

(b) Data preparation and input: direct entry key-to-tape device

**Fig. 2-52** Comparison of data preparation and input with punched cards (top) and key-to-tape device (bottom)

*Key-to-tape.* In preparing data on a key-to-tape device, an operator sits at a keyboard linked to a magnetic tape drive. Each keystroke is translated into magnetic tape codes understandable to the computer, and is magnetically recorded as spots on the tape.

The key-to-tape data recorder actually has a small core storage area, which makes it possible for the operator to correct errors as he or she goes. As each character is typed, it is converted to magnetic tape code and stored in core storage. Then, when a complete record has been keyed in and stored, the entire record is recorded on the tape at once.

This allows an operator to backspace and retype an incorrect entry during the initial phase before the complete typed-in record is recorded on tape.

Again, key-to-tape offers many advantages over punched cards or paper tape for input. Besides speeding up both the data preparation operation and the input operation, there are other advantages. The format of the input data is not limited to 80 columns, as in a card, and recorded data cannot get out of sequence as they can with cards. Elimination of punched cards saves the cost of cards, manual handling, and some keypunch operator and training expense.

*Key-to-cassette and key-to-cartridge.* There are occasions when data must be recorded where they originate, and it is not practical or economical to install a large, expensive key-to-tape data recorder. In

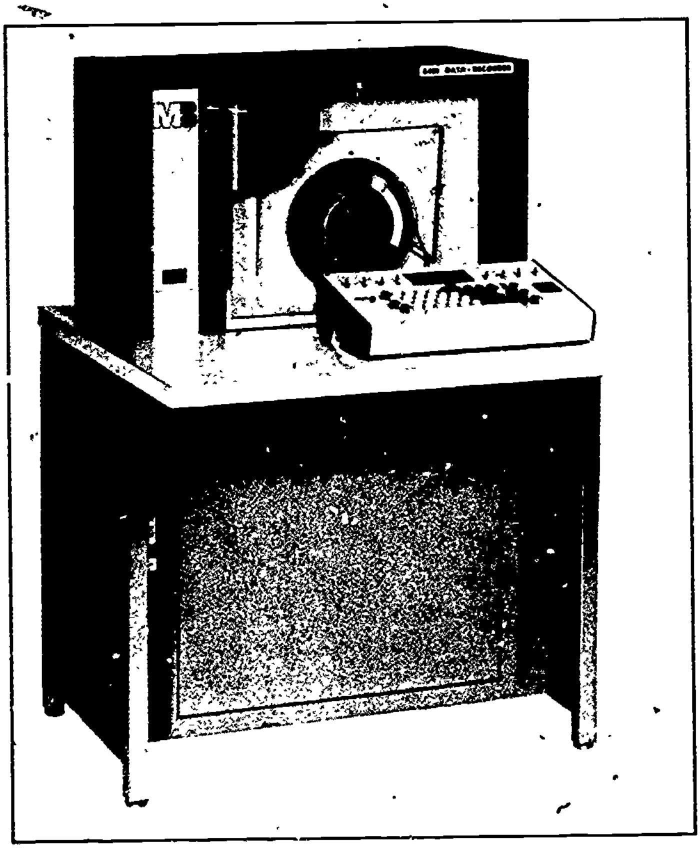


Fig. 2-53 A Mohawk Data Sciences 6401 data recorder

such cases, a key-to-cassette or key-to-cartridge recorder may be used. The recorder may be connected to a keyboard, such as a teletypewriter, and data are recorded from the keyboard directly onto a small (2½ inch by 4 inch) magnetic cassette tape or on a cartridge. Later, the cassette or cartridge may be read on to a full-size magnetic tape for computer processing. A large organization, for example, may have small regional offices where data are recorded on a cassette or cartridge. The cassette or cartridge is then sent for processing to the main office either through the mail or over telephone lines. Or a cash register can be equipped with a magnetic tape cartridge which records all sales information. The cartridge is then sent to a central location at the end of each business day for processing.

Some mini-computers (small computers) accept magnetic tape as input directly from cassettes.

The ¼-inch-wide tape, when compared to the traditional ½-inch-wide tape, offers several advantages. It is more convenient and economical to handle and store a complete batch of information on a single cartridge, which holds 28,000 characters, or on a cassette, which holds about 200,000 characters.

***Key-to-disk.*** A key-to-disk system is a data preparation and input system which is even more sophisticated than a key-to-tape system.

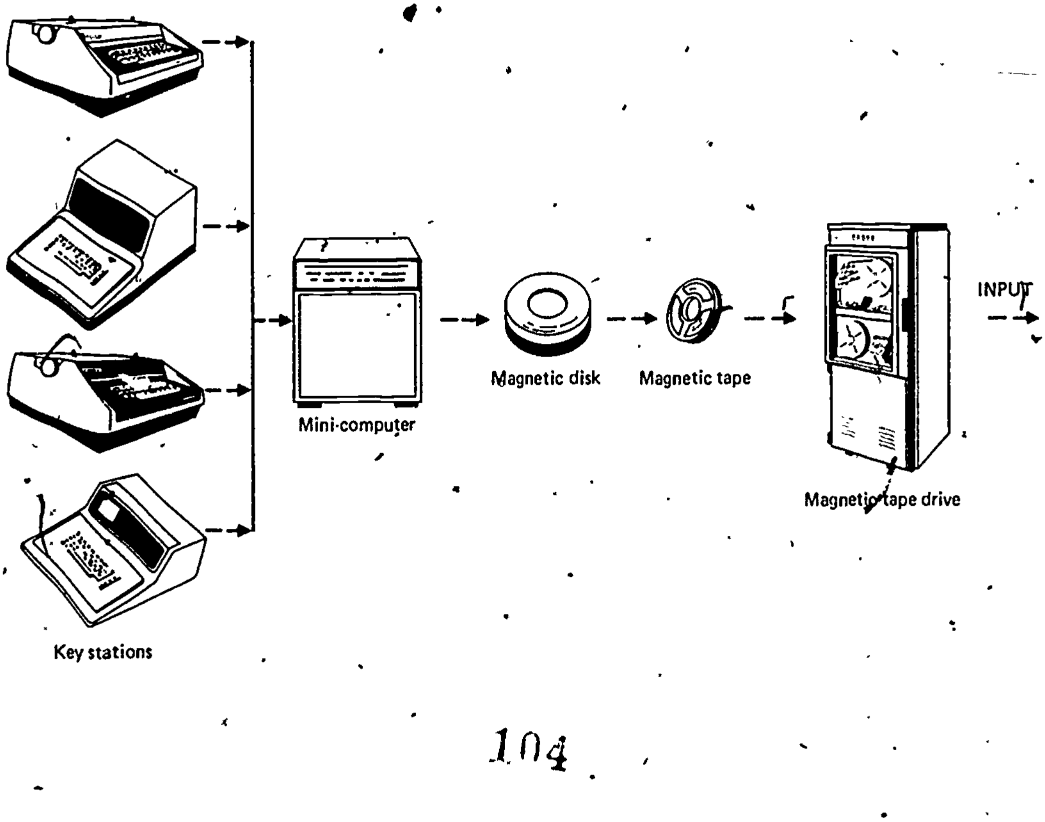
In a direct-entry key-to-disk system, a single magnetic disk or disk pack can be connected simultaneously to dozens of "key stations" with operators keying in data at each station. A small-size computer is also connected to the disk, for the purpose of editing data from the key stations before they are recorded on the disk. Once the disk is loaded, it can be read out on magnetic tape for computer processing, or the data may be processed directly from the disk, which is usually the case.

Error rates are decreased into a key-to-disk system, since many systems have a character-display panel at each key station which displays the last character typed or even the entire record. Thus the operator can see errors and make instant corrections.

A disk system with a number of key stations can replace twice that number of keypunch machines. These systems are expensive now (up to $125,000), but as the cost is decreasing, the sales are increasing rapidly. Although they are expensive to install and operate, key-to-disk systems are more efficient in meeting the data preparation demands of installations where large volumes of data are prepared for input.

**Fig. 2-54** Key-to-cassette device (left). A variety of cassettes are available (right).

**Fig. 2-55** Flow from keyboard through disk to input



## THE OUTLOOK FOR THE FUTURE

Direct-entry data preparation devices are becoming increasingly popular. Some predictions indicate that the keypunch will be a relic of the past in a few years, while others see companies as reluctant to switch from using the keypunch as the primary data preparation device.

Whether direct-entry devices replace the keypunch or merely supplement it, certainly data clerks will have to be prepared to be retrained perhaps several times during their careers, as new data preparation devices are adopted. After a nationwide survey of 1265 data processing installations,* it was reported that "Installations are going to remote terminals and key-to-disk. As a result they are promoting keypunch operators to more comprehensive jobs. Keypunch operators who do not want the added responsibility are often being let go." Data clerks, then, can look forward to careers which become more interesting as technology improves. On-the-job training will equip most of the interested data clerks with the skills necessary to operate the newest devices.

Check your understanding about direct-entry data preparation devices by answering the following questions.

# Check your understanding

1. Direct-entry data preparation devices were developed

   a. to eliminate the keypunching bottleneck.
   b. just after punched cards.
   c. as a way to speed up keypunching.
   d. about the same time as modern computers.

2. A "key-to-tape" machine

   a. allows the operator to read cards onto tape.
   b. allows the operator to key data directly into the computer.
   c. requires the operator to record a new tape if an error is made.

* Conducted in 1973 by Philip H. Weber Salary Administration Services, A.S. Hansen, Inc., 1080 Green Bay Road, Lake Bluff, Illinois 60044.

d. records magnetic spots directly on magnetic tape after temporary storage in the machine.
e. saves time and money compared with keypunching cards.
f. speeds up not only the data preparation operation but also the input operation.

3. Key-to-cassette or key-to-cartridge recorders usually are used

a. in a key-to-tape data recorder.
b. when data must be recorded "on the spot."
c. when immediate processing is needed.
d. when greater storage capacity is needed.

4. A key-to-disk system

a. decreases error rates.
b. usually has more than one operator "key station."
c. usually is connected to more than one disk pack.
d. both a and b.
e. both b and c.

5. The keypunch operators of the future

a. will be replaced by direct-entry systems.
b. should be prepared to learn more sophisticated methods of data preparation.
c. will be let go.
d. will be promoted.

# 1234567

# Input and output devices

## INPUT DEVICES

### Punched card reader

Once cards have been punched by a keypunch operator, they are ready to be read into the computer by an input device—in this case, the input device we will consider will be the punched-card reader.

The card reader is designed to recognize holes punched in a card and to transmit that code to the CPU. It will transmit only valid

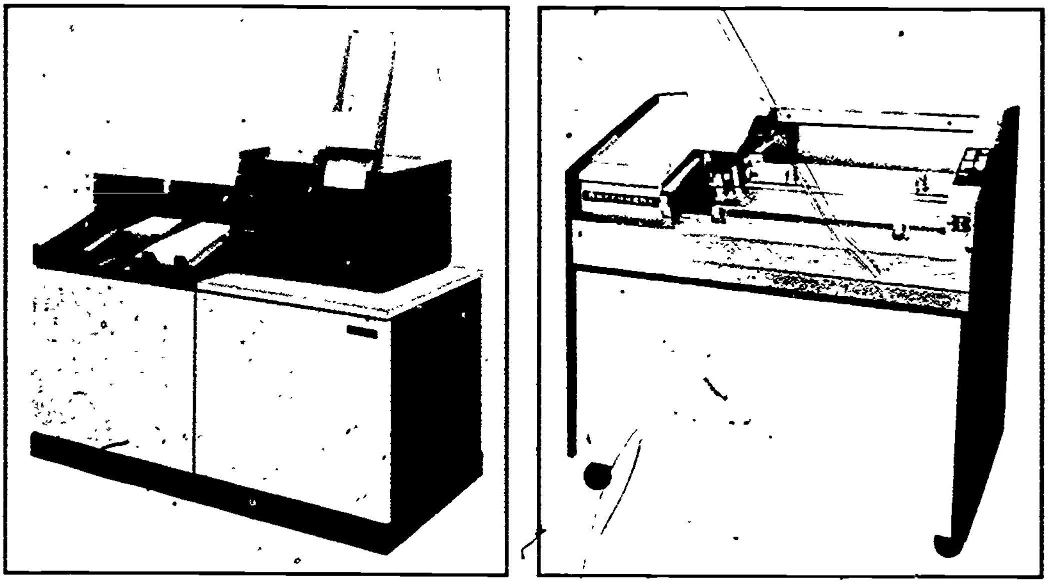


**Fig. 3–1** Two typical punched card readers. An IBM 3505 card reader (left) and a Burroughs 9111/9112 card reader (right)

characters to the CPU. Usually a card reader has a built-in device to detect invalid characters and to discontinue reading when one is detected. When this happens, the operator must correct the error before any further reading can be done. (An example of an invalid character is five punches in a single column. The card reader cannot interpret this "code.")

There are two basic types of card readers in use: the machine that reads with a brush-type mechanism and one that reads with a photoelectric cell.

In a brush-type reader, a reading brush on top of the card makes contact with a roller beneath the card every time a hole is detected.

The photoelectric cell equipped reader uses a light-sensing device. The card passes beneath a light, holes in the card allow the light to shine through them, and this light establishes contact with photoelectric cells.

Punched card readers in use today can read cards at speeds of from 300 cards per minute to 1600 cards per minute. (That's more than 20 cards each second!) Imagine how many keypunch operators would be needed to keep one high-speed card reader operating constantly.

You can see that, with the speeds of most such input devices, the data preparation step is frequently the "bottleneck" that slows down the entire flow of information through the computer system. If the system is to operate efficiently, a number of keypunch operators are needed to supply input for a single card reader.

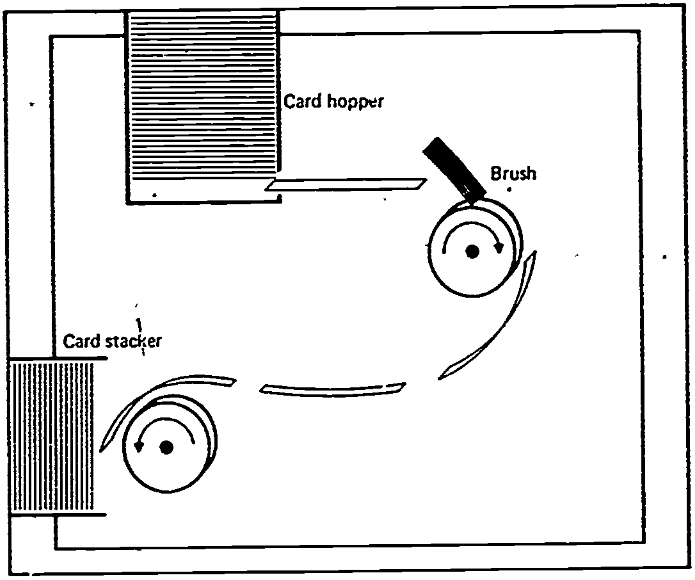


**Fig. 3–2** A brush-type card reader reading a punched card

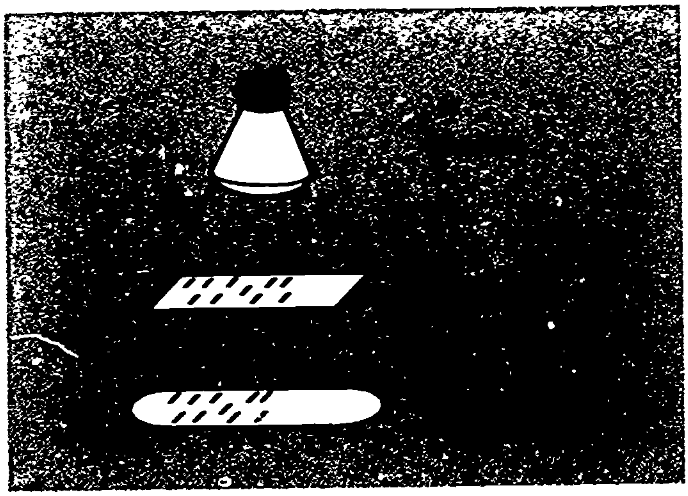


**Fig. 3–3** A photoelectric cell-type card reader reading a punched card

## Optical page reader

The optical page reader (or "optical scanner") is designed especially to read optical scan sheets such as those mentioned on page 79. Using an optical scanner, such as the one pictured in Fig. 3-4, thousands of test answer sheets can be automatically and accurately scored and the results recorded in the same time it would take to score one answer sheet by hand.

Sometimes the optical scanner reads the data from the optical scan sheet directly into the computer for processing. In other instances, the data may be read onto magnetic tape which is saved for later processing.

## Paper tape reader

Until now, you probably have been using the teletypewriter strictly in the data preparation mode—that is, in "local mode"—for

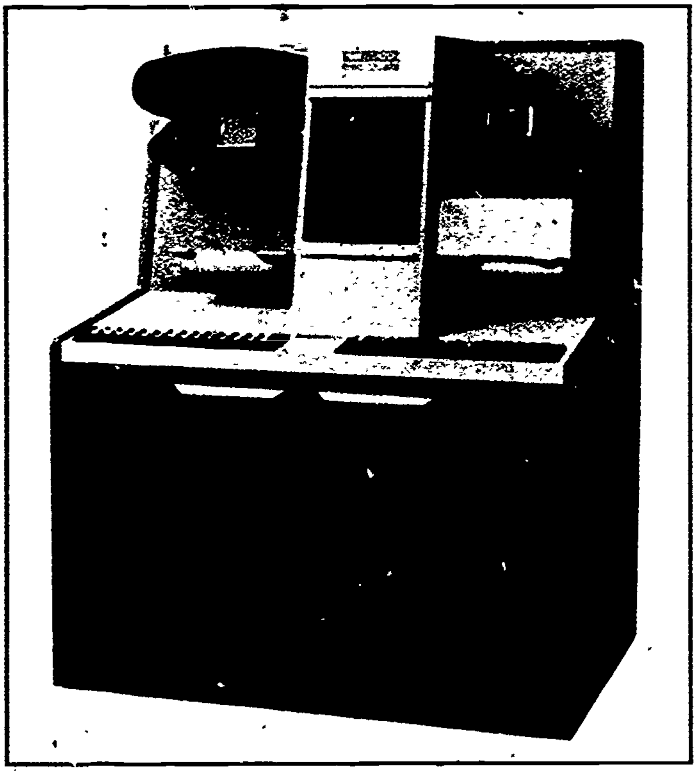


Fig. 3-4 A Westinghouse W-300 Optical Mark Reading System

punching paper tapes. This mode is often called "off-line"—that is, not connected to the computer. Once you have a prepared paper tape, however, you can convert the teletypewriter to "on-line" mode by connecting it directly to the computer and using the Tty's paper tape reader to input the data from your paper tape.

Teletypewriters (or Tty's) most often make connections with a computer by a conventional telephone line. The telephone number for the computer is dialed and, when the answering high-pitched sound is heard, the telephone receiver is placed on the coupler attached to the Tty. The coupler also emits a high-pitched sound, which is picked up by the receiver and transmitted to the computer. In that way, the connection between the two machines is made, and they can now transmit information to each other in code in the form of patterns in their respective sounds. With the telephone connection made, the teletypewriter's paper tape reader can be used to input data from your tape.

With a teletypewriter in "on-line" mode you can input data also by typing at the keyboard.

But, since the paper tape reader can read a prepunched paper tape at a speed of ten characters per second (or ten rows of code-holes per second), it is obviously considerably faster than typing—especially if you hunt-and-peck! Saving time also saves money, if you are being charged for the length of time the teletypewriter is connected to the computer.

There are many models of paper tape readers other than the one found on most teletypewriters. Some are used as parts of terminals at a distance from a central computer and others are used as input devices connected directly to a computer. Fig. 3-5 shows a Digitronics paper tape reader. It is a high-speed reader often used with mini-computers.

## Magnetic tape drive

Another input device common to computer systems is the magnetic tape drive. It reads input data from magnetic tape and transfers it to the computer. As you know, magnetic tape used in connection with computers works something like regular magnetic recording tape. Likewise, the magnetic tape drive unit works very much like a regular tape recorder. In fact, just like the average tape recorder, the magnetic tape drive can both "record" (or "write") data *onto* magnetic tape and "read" data *from* the tape.

The magnetic tape itself is always stored and handled on reels, like those pictured in Fig. 3-6. Data is carried on the magnetic tape in code in the form of magnetized spots rather than in the form of holes as in cards or paper tape.

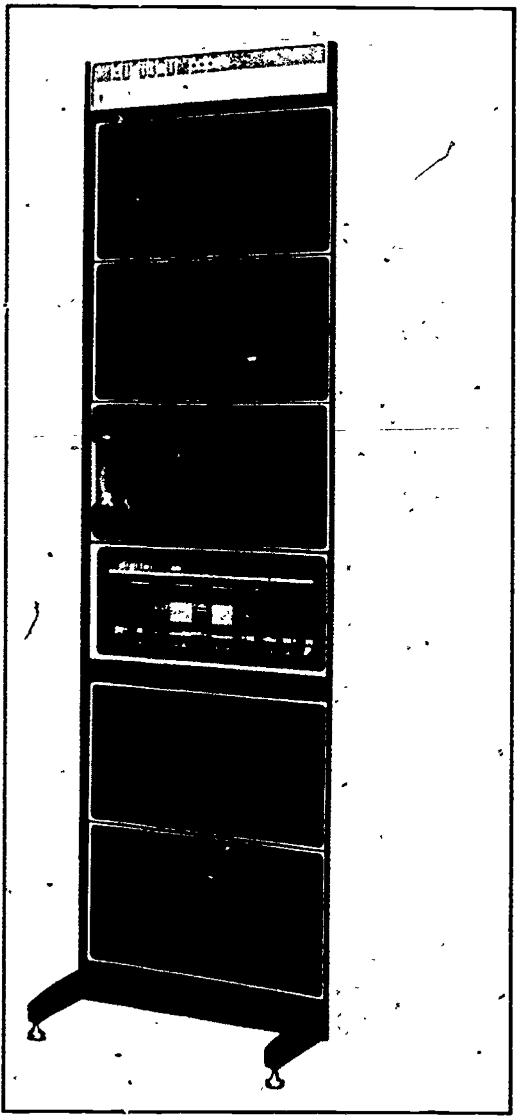


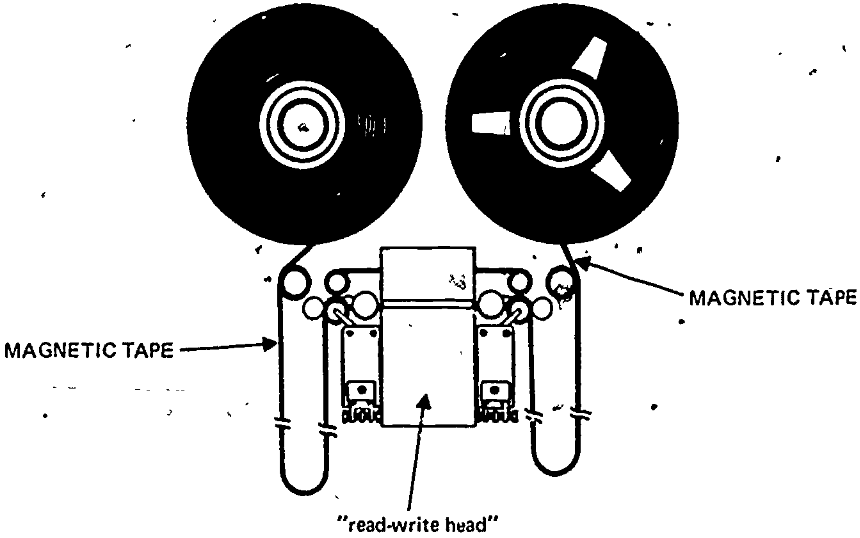
**Fig. 3-5** A Digital high-speed paper tape reader

How do the magnetic spots get "written" onto the tape? For the answer, remember what we said before about the magnetic tape units being able both to record *onto* and to read *from* tapes, and then look more closely at the illustration of the magnetic tape unit in Fig. 3-6.

The magnetic code is written onto the tape by the read-write head located at the center of the unit. It is called a "read-write" head because it does both the "writing" onto tape and the "reading" from tape for input into the computer. Inside the head there are usually two sets of magnetic coils which can either sense the presence of magnetized spots (that is, "read" them), or magnetize spots on the tape's surface ("write" them).

You will remember that there are special machines that can be used to record the magnetic spots directly on tape from a keyboard. Without a key-to-tape machine, however, it still is common for computer systems to use magnetic tape. In this case, there is no manual keyboard available. The read-write head is run electronically from the computer itself. So, if a program or some data needs to be put onto magnetic tape, it is first put into the computer from punched cards, paper tape, or any other medium. The computer is then simply instructed to output the same material on magnetic tape through the tape drive's read-write head.

**Fig. 3–6** Magnetic tape read-write mechanism



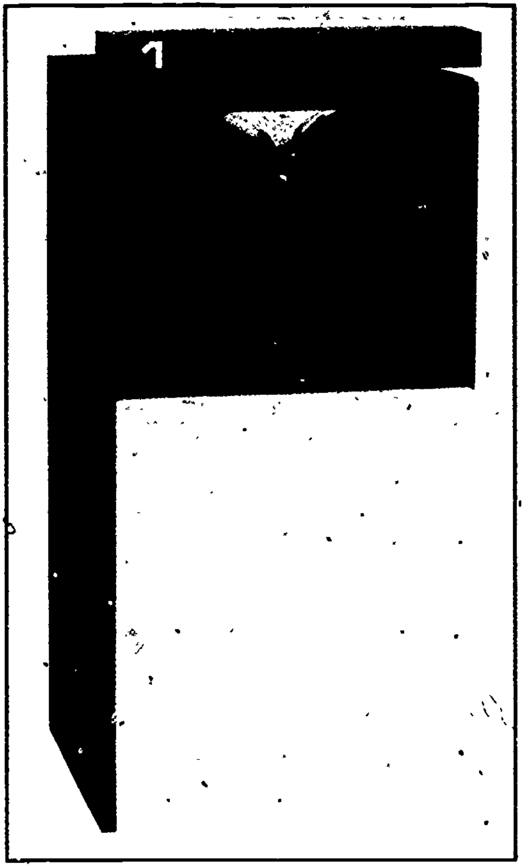


**Fig. 3–7** A typical magnetic tape drive

Computer centers often have a library which is comprised of master magnetic tapes encoded with often-used programs and data. When a computer run calls for the input of material from these tapes, the operator simply locates the appropriate tape in the tape library and mounts it on the magnetic tape drive. When it needs the material, the computer can then activate the tape drive's read-write head to "read" the coded spots and transfer the data electronically into the computer for use. The computer itself has complete control over activating the tape drive to input (or "access") material. But, the computer does so only according to instructions. These instructions come from the program being run or from the special instruction cards for the run which are given to the operator by the programmer.[a]

[a] These cards are most often referred to as "job control cards." They are commonly used to tell the computer how to prepare for a run.

Most magnetic tape units look something like the unit pictured in Fig. 3-7. They are set up and threaded very much as regular tape recorders are. Instructions for setting up individual tape drives are always available in the manuals accompanying the given unit.

When a complete program or batch of data (called a "data file") is being input all at once by a magnetic tape drive, you can observe one of the things which make magnetic tape and the magnetic tape drive so useful for inputting data into the computer—their speed!

There are two input devices, however, which are even faster than a magnetic tape drive. They are the magnetic disk drive and magnetic drum drive.

## Magnetic disk drive

The magnetic disk drive is used to input data from magnetic disks. You will recall that data on a disk are in the form of magnetic spots, which are recorded and read by read-write heads. These heads are situated in the drive unit so that there is one read-write head for each disk surface in the regular disk pack, as shown in Fig. 3-8.

The access arms can be moved in or out so that the heads are at any chosen track position. The head for the desired surface can then be activated to read the magnetically coded spots (representing data) on the correct track of that surface.

The read-write heads of the disk drive are run electronically by the computer. So, to record new programs or data onto magnetic disks, the material must first be fed into the computer on punched cards, paper tape, or other media. The computer can then be instructed to output the same material on magnetic disk through the disk drive's read-write heads.

Large computer centers often have considerable material stored on magnetic disks, including their most frequently used programs and data. When a computer run calls for input which is stored on a disk pack, the computer operator simply removes the correct pack from the storage room. Then it is mounted on the input device (the disk drive) very much as a stack of records would be mounted on a record player.

When the computer needs the data, it directs the drive unit to move the access arms on the read-write assembly to the correct track position and to activate the read-write head for the correct surface to "read" the coded data into the computer. As in the case of the

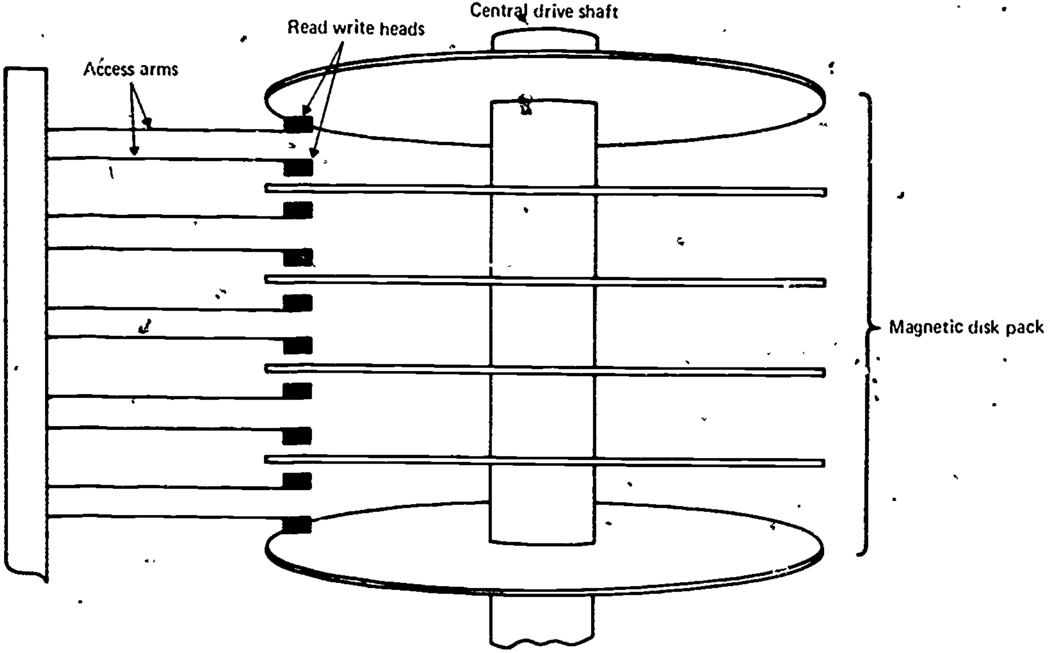


**Fig. 3–8** Illustration of a mounted disk pack with read-write heads in position

magnetic tape drive, the computer alone can access (or locate) data on disks through the disk drive. The computer does so according to instructions given it in the program it is running or on the job control cards for the run.

From the description of accessing data above, you can see that the capability for random accessing is the feature of the disk pack which makes it a faster input medium than magnetic tape. To select out any given data on a magnetic tape, the entire tape must be run, foot by foot, through the read-write head until the place of the correct data is reached. To locate the given data on a magnetic disk, however, the access arms on the read-write assembly need only to be moved to the correct track position, and the needed head activated. Can you see that the programs and data recorded on the hundreds of tracks on the ten disk surfaces in the pack can be reached (or "accessed") immediately with just a shift of the read-write head assembly?

If a large computer center has a tremendous amount of data which it uses almost constantly, you can see that the computer operator would be better off if the data were stored in disk packs rather than on

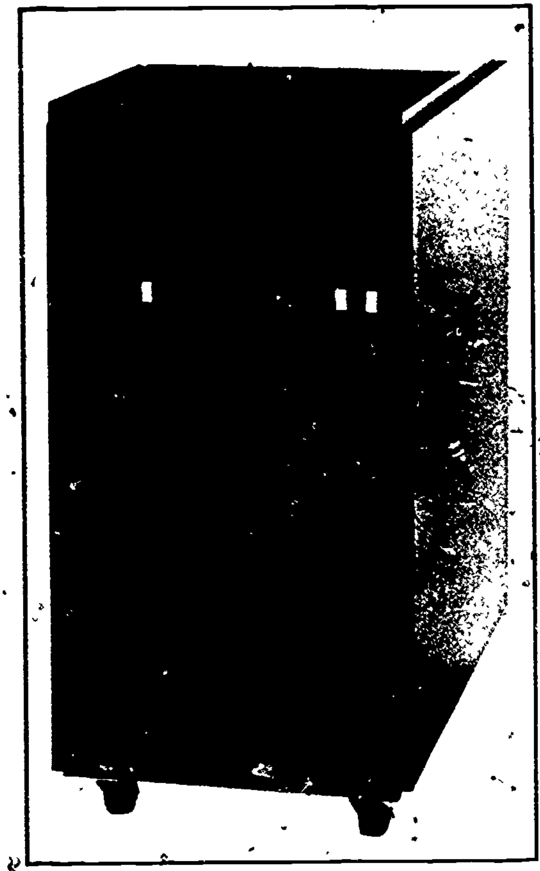


Fig. 3-9 A typical disk drive

magnetic tapes. Still, disk packs would have to be located, mounted, and dismounted. Another input medium often used in large computer installations for the rapid input of continually used programs or data is the magnetic drum.

## Magnetic drum drive

As you have learned, a magnetic drum is a metal cylinder coated with magnetic material. Data are recorded as magnetized spots in circular tracks or bands around the surface of this drum.

Data are always recorded onto and read from magnetic drum by the read-write assembly in the magnetic drum drive unit. This assembly is actually a series of read-write heads, one for each band on the drum. The heads are positioned near the surface of the drum to record or read magnetic spots as the cylinder rotates. If you examine Fig. 2-51, you will see that there is one read-write head for each band. This makes it possible for all the bands on the drum to be scanned simultaneously. Any particular data, therefore, can be located within one swift revolution of the drum, and it is this feature of the drum that makes it a faster medium than the disk and, for certain kinds of jobs, more convenient.

From the description above you can see that the magnetic drum would be a harder medium to carry around, store, mount, and dismount than a disk or magnetic tape. This is true not only because it is often very large but because its entire cylindrical surface, which carries the data, is entirely exposed and highly delicate. For these reasons, the magnetic drum is ordinarily mounted permanently on the drive unit, where it is protected by a vacuum-tight cover. Because the magnetic drum is left always in position, computer centers need to make certain that they are installing a drive unit with a drum sufficiently large to carry all the data they need to have on it.

Once a magnetic drum is encoded with a center's most constantly used programs and data, it is ready on the magnetic drum drive for input at any time. When a particular program or data file on the drum is needed for a run, the computer directs the read-write heads to scan the tracks for the correct material as the drum rotates at high speed.

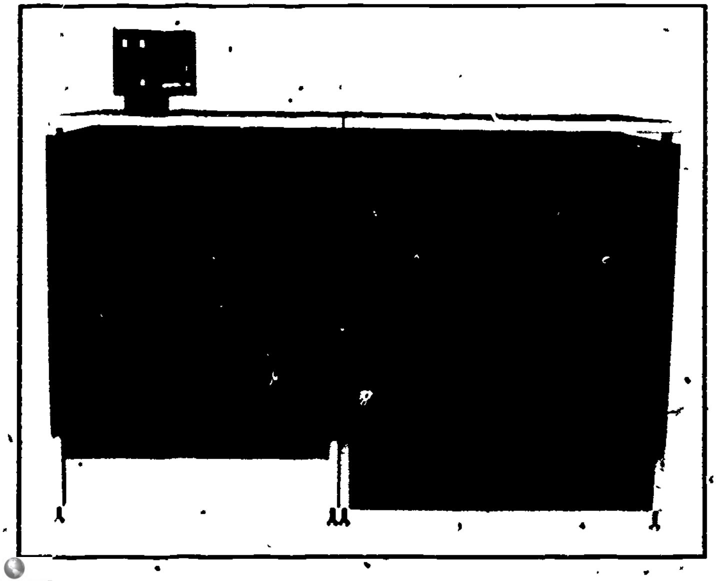


Fig. 3–10 A magnetic drum drive

Once it is located, the coded material is "read" by the correct read-write head and is electronically transferred into the computer for use. The data on a magnetic drum can be accessed for input only by the computer, according to instructions given it by a program, job control cards, or the console typewriter. (The console typewriter is discussed in greater detail later in the chapter.)

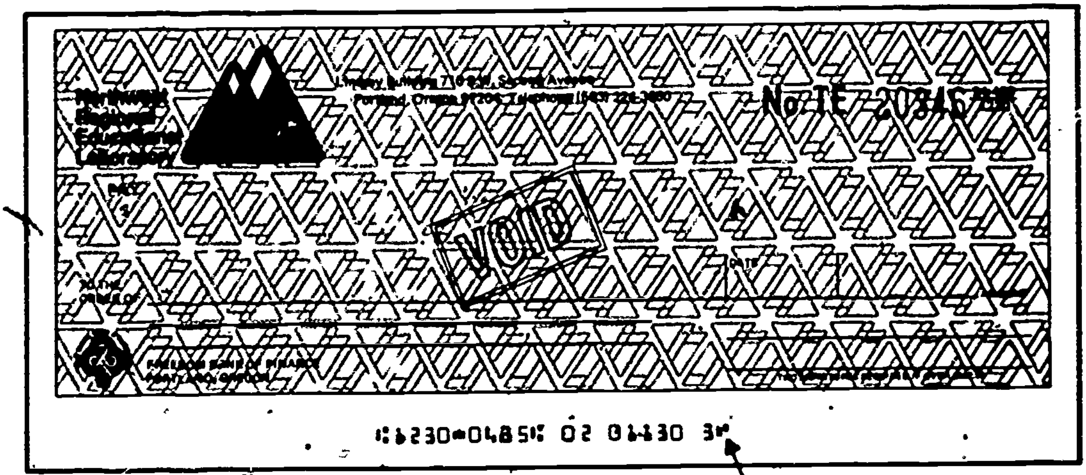
## Pattern recognition devices

Another challenge to the keypunch/card monopoly in data preparation and entry is the use of devices which recognize information in printed or written form. One such device uses magnetic-ink character recognition (MICR), and another is an optical character recognition (OCR) device.

No doubt you have seen magnetic ink characters on personal checks. If you don't have a checking account of your own, look at one of your mother's or father's checks. The bank number and the checking account number will both appear in oddly shaped numbers and symbols at the lower left edge of the check. These are magnetic-ink character recognition (MICR) characters. They are printed on the check with magnetic ink.

MICR was developed especially to help banks process all of the various sizes and shapes of checks that go through a bank. Magnetic

Fig. 3-11 Magnetic-ink characters of a check



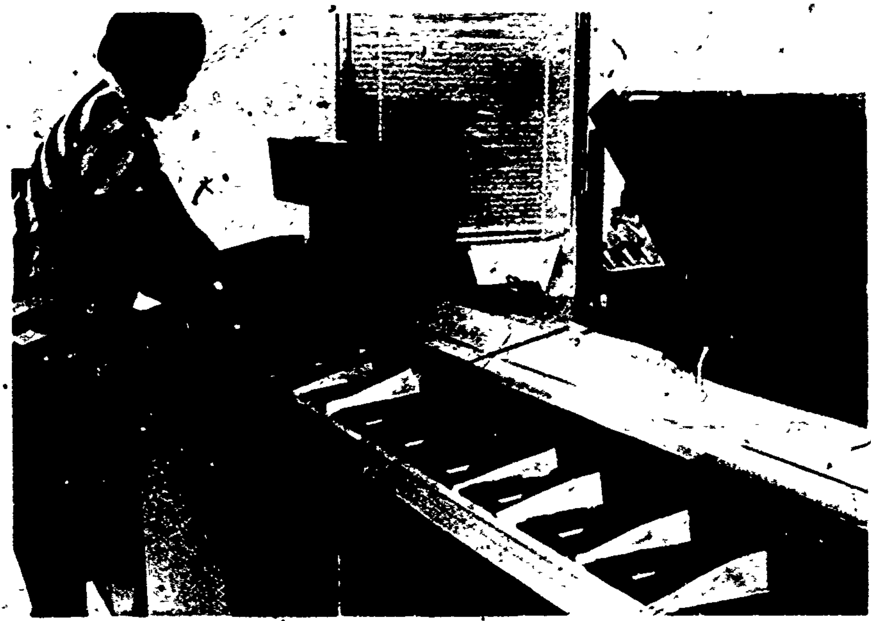
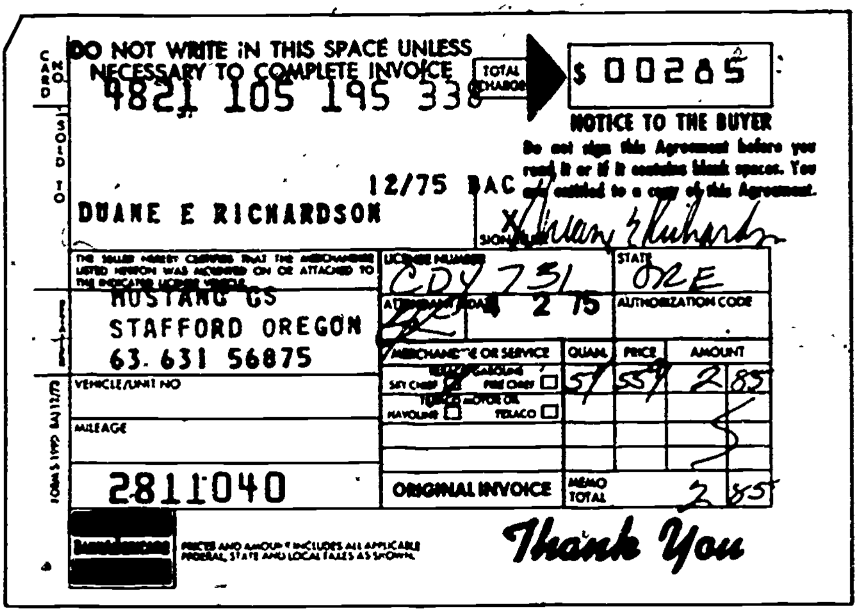


Fig. 3–12 A typical magnetic character reader-sorter

Fig. 3–13 A gas credit card receipt with OCR numbers



CARD NO.

DO NOT WRITE IN THIS SPACE UNLESS NECESSARY TO COMPLETE INVOICE

4821 105 195 338

TOTAL CHARGE $ 00285

NOTICE TO THE BUYER

Do not sign this Agreement before you read it or if it contains blank spaces. You are entitled to a copy of this Agreement.

SOLD TO

12/75 BAC

DUANE E RICHARDSON

SIGNATURE X Duane E Richardson

THE SELLER HEREBY CERTIFIES THAT THE MERCHANDISE LISTED HEREON WAS ACQUIRED ON OR ATTACHED TO THE INDICATED LICENSE VEHICLE

MUSTANG GS
STAFFORD OREGON
63 631 56875

LICENSE NUMBER CDY 751

STATE ORE

ATTENDANT DATE 2 75

AUTHORIZATION CODE

| MERCHANDISE OR SERVICE | QUAN. | PRICE | AMOUNT |
|---|---|---|---|
| TEXACO GASOLINE<br>SKY CHIEF ☑ FIRE CHIEF ☐ | 5 | 55 9 | 2 85 |
| TEXACO MOTOR OIL<br>HAVOLINE ☐ TEXACO ☐ | | | |
| | | | |
| | | | |
| ORIGINAL INVOICE | MEMO TOTAL | | 2 85 |

VEHICLE/UNIT NO

MILEAGE

2811040

BANKAMERICARD

PRICES AND AMOUNT INCLUDES ALL APPLICABLE FEDERAL, STATE AND LOCAL TAXES AS SHOWN.

Thank You

ink makes it possible to use the original check as computer input, with no further data preparation. The MICR characters are readable by humans as well as machines. Although the MICR characters meet the needs of banks and other financial institutions, they cannot be used in many data processing problems, since there are no alphabetic characters, only numbers.

The input device which reads these MICR numeric characters is called the magnetic character reader-sorter. One is shown in Fig. 3-12. The reader-sorter can read MICR characters from checks of many different lengths, widths, heights, and thicknesses. It either records the data on magnetic tape for later processing or transmits the information directly to the CPU for updating customers' accounts. As it reads information from the checks, the reader-sorter also sorts them by account number into its various pockets. All of this takes place at a rate of about 1600 checks per minute.

Optical character recognition (OCR) eliminates keypunching and allows data to be prepared simply by typing or even hand-writing the data. One common use is in credit cards: the next time your parents charge a tank of gas with a credit card, examine the receipt. The card number and amount of purchase will be printed in specially-shaped numbers, as shown in Fig. 3-13.

The input device, an optical character reader, can read these numbers directly into the CPU or onto tape or cards for later processing and customer billing. Some optical character readers can read typewritten characters from a page, or even hand-written numbers. For example, data might be prepared on the spot by utility meter readers or salespersons, and read directly into the computer by an optical character reader.

Another type of optical reader will accept and read plastic ID cards or badges to admit personnel to a high-security area, for instance. Such a device is shown in Fig. 3-14.

The mark-sense card reader we discussed earlier is actually another type of optical character reader. It reads pencil marks placed in predetermined positions on a card or paper document. Similar types of readers are used to automatically score the millions of answer sheets from the standardized tests (achievement tests, for example) that students take every year.

Optical character readers, or "optical scanners" as they frequently are called, vary in speed and costs. Some scanners read only 70 characters per second, and others operate at speeds of up to 2400 characters

**Fig. 3–14** Example of an optical ID card reader system

per second. Predictably, cost of optical character readers goes up with increased reading versatility and speed.

Now check your understanding about input devices.

## Check your understanding

1. A punched card reader

   a. may be brush-type or photoelectric-cell type.
   b. can read cards much faster than people can keypunch cards.
   c. is the most common input device found in computer centers.
   d. all of the above.

2. An optical page reader

   a. is commonly used to score and record test answer sheets.
   b. is designed especially to read optical scan sheets.
   c. both a and b.
   d. neither a nor b.

3. A teletypewriter, can usually be used for

   a. data preparation.
   b. input.
   c. both a and b.
   d. neither a nor b.

4. "Local mode" means

   a. off-line.
   b. on-line.
   c. directly connected to the computer.
   d. input mode.

5. A teletypewriter is converted to an "on-line" device by

   a. the paper tape reader.
   b. connecting it to the computer by a telephone line.
   c. using it as an electric typewriter.
   d. using it in local mode.

6. An important input device on a teletypewriter is the

   a. paper tape punch.
   b. paper tape reader.
   c. keyboard.
   d. both a and b.
   e. both b and c.

7. A magnetic tape drive unit can

   a. read data from magnetic tape.
   b. record data onto magnetic tape.
   c. be used only for input.
   d. both a and b.
   e. both a and c.

8. The magnetic tape drive's read-write head can

   a. read data from magnetic tape.
   b. record data onto magnetic tape.
   c. be used only for input.
   d. both a and b.
   e. both a and c.

9. The magnetic tape drive is activated by

   a. the computer.
   b. a human operator.
   c. a key-to-tape machine.
   d. a data clerk.

10. Magnetic disk and magnetic drum units are

   a. faster than a magnetic tape drive because they are random access devices.
   b. faster than a magnetic tape drive because they are serial access devices.
   c. slower than a magnetic tape drive because they are random access devices.
   d. slower than a magnetic tape drive because they are serial access devices.

11. A six-disk pack will need

   a. six read-write heads.
   b. twelve read-write heads.
   c. ten read-write heads.
   d. none of the above.

12. Magnetic-ink character recognition (MICR) and optical character recognition (OCR) devices

   a. recognize information in printed or written forms.
   b. bypass the keypunching operation.
   c. can read data in the same form as humans can.
   d. all of the above.

## OUTPUT DEVICES

Data go in—
information comes out

The function of output devices is, naturally, to output the information produced by the computer in a form that is understandable to humans. Perhaps at this point we should clear up a distinction between two words we have used frequently, "data" and "information." *Data* are the raw material to be processed by the computer, and *information* is the meaningful result of that processing.

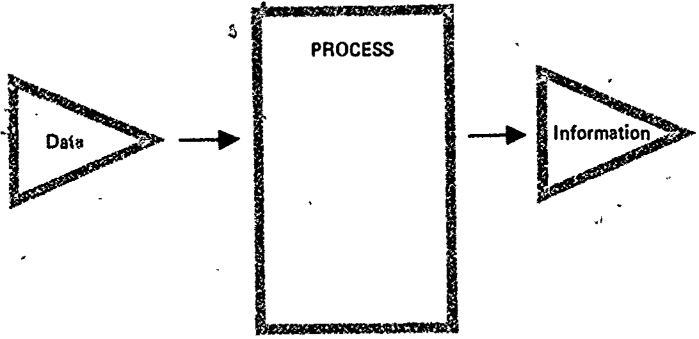
Data in unprocessed form are not very meaningful to human beings. When data are arranged and presented in a meaningful manner, they are called information. The number "54" is an item of data but is not, by itself, information. If your teacher says to you, "You scored a 54 on your test," you still don't have any more knowledge than you had before unless you can relate that to other data or information—or unless the teacher gives you more information. If you know that the test consisted of 100 very easy items, does your score of 54 now become "information"?

If you know that the test consisted of 60 very difficult items, is a score of 54 information? Or, is it information if your teacher tells you your score was the highest in the class? Of course! If the data "54" is related to other data in a meaningful way, it becomes information.

Output from a computer system usually consists of data that have been combined, rearranged, calculated, and analyzed to produce information. For example, if the input data are all the semester grades of all the students in your school, what might the output information be?

The grade lists submitted by each teacher are data, and the rearranged grades are information. Report cards, honor roll, and failure list provide a meaningful way of reporting data. Report cards are meaningful to each student and to the student's parents. The honor roll tells everyone who were the highest achieving students in the whole school. The failure and incomplete lists tell teachers and administrators who needs more help and encouragement. They identify students who are failing every class, so that counselors can talk to the students and work out ways to help them improve.

**Fig. 3–15** Data are the raw material to be processed; information results from processing.



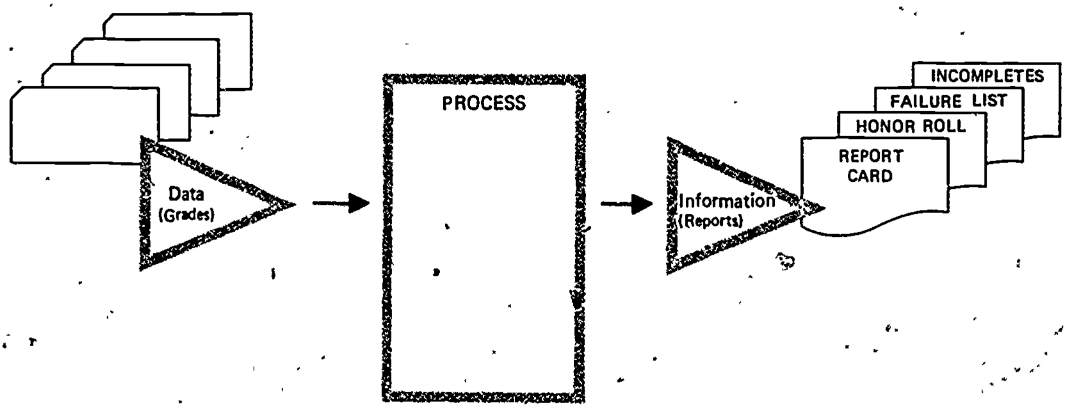


Fig. 3-16 Data become information through processing.

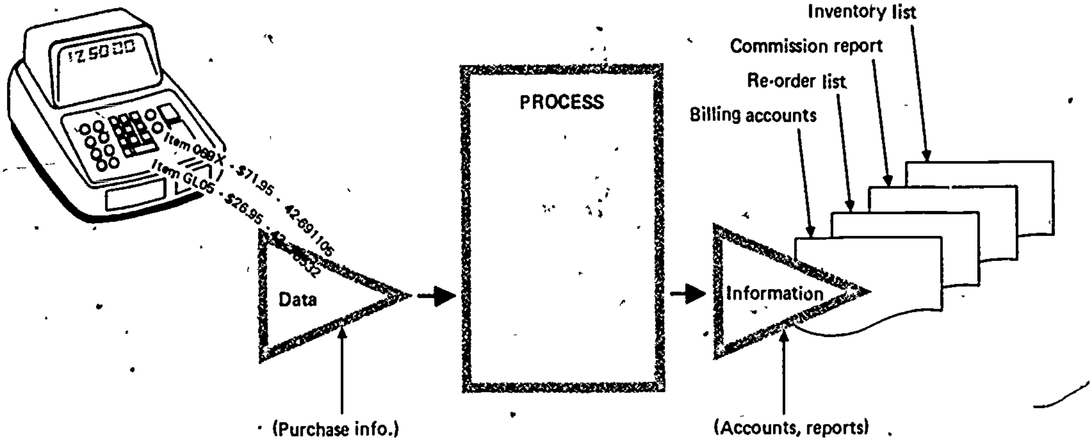


Fig. 3-17 Billing data can result in various kinds of information.

Consider the chain department store that inputs data concerning sales and charges to a central comp.ter every day. After being processed, calculated, and recombined, the output could provide valuable and necessary information to many decision makers. Think of all the information that could be obtained just from data about sales (color, stock number, price, clerk's department number and identification number, and charges):

- A list of items by color and stock number that need to be restocked from the warehouse.
- A list of items that need to be reordered to maintain the inventory in the warehouse.

- A report for the payroll department showing the amount of commission each clerk is entitled to, based on volume of sales.
- A summary of total sales for the day, by department, for the managers of that particular store and for the central headquarters.
- A list of charges, for the billing department.

A variety of output devices can be used to produce all of this information. A visual display on a terminal with a TV-like screen might be sufficient for some users, while others might need the information punched on cards, or printed on paychecks, invoices, or as a report.

Let's examine some of the more commonly used output devices in greater detail.

## Printers

In our paper-oriented society, the computer printer is the most indispensable part of the data processing operation. A printer translates the computer's electronic way of representing information into information that is readable by human beings. There are two common kinds of printers: the *character printer* and the *line printer.*

Character printers print a single character at a time, one after the other, the way a typewriter prints. An example of this kind of printer would be the teletypewriter, or any other keyboard device. Even a terminal with a TV-like display screen is often a character-at-a-time printer, although it may be faster than a typewriter. Most character-at-a-time printers are dual devices, used for both input and output.

The most commonly used output device is the line printer. It is called this because it prints all characters on a given line simultaneously. When you consider that a line of print can have 100 or more characters in it, you can readily see that a line printer operates at a much greater speed than typewriter-like devices which can only print one character at a time. In fact, an average line printer can print at a speed of ten complete lines per second or faster. (How many "characters per minute" can be printed at a speed of ten 100-character lines per second?)

The printer is always under the control of the computer which sends it electronic impulses of information. These impulses cause the printer to print out the information using the regular alphabet, num-

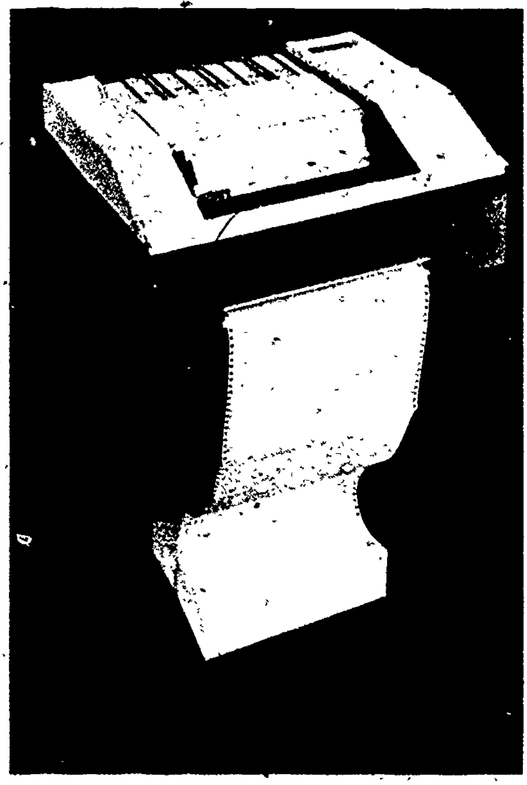
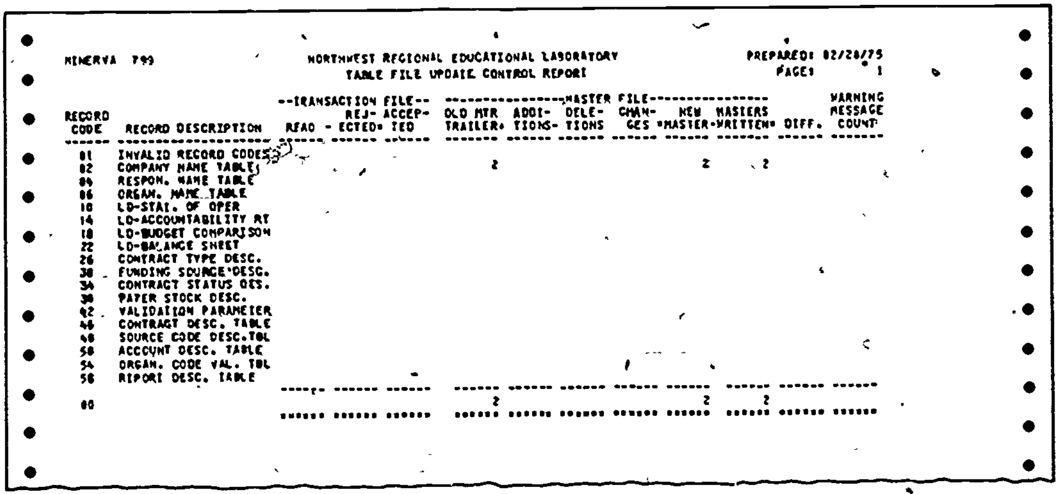


Fig. 3-18 Line printers print a full line at a time.

bers, or characters that people can read. The printout paper used by the conventional printer comes in various size rolls with holes at both edges. The paper on the roll is continuous, but the page lengths are marked off by tiny perforations for easy tearing later.

The printer not only will print as directed by the computer, but it can be set up to detect when the perforations marking a page's end have been reached. It will automatically skip over the perforations and begin printing again on a fixed line (so many lines down from the perforations) on the next page. The printer can also detect when the end of the paper roll has been reached and signal the operator. At this point the operator must load a new roll.

There are two basic types of line printers: impact printers (which print using a type bar or wheel pressed against the paper), and nonimpact printers (which form an image by chemical or other nonimpact means). Almost all printers use an impact mechanism and produce output which looks very much like that shown in Fig. 3-19.



```
MINERVA  799                NORTHWEST REGIONAL EDUCATIONAL LABORATORY                  PREPARED: 02/28/75
                                   TABLE FILE UPDATE CONTROL REPORT                    PAGE:          1

                                --TRANSACTION FILE--  ---------------MASTER FILE---------------          WARNING
RECORD                                REJ-  ACCEP-  OLD MTR  ADDI-  DELE-  CHAN-   NEW  MASTERS         MESSAGE
 CODE   RECORD DESCRIPTION    READ - ECTED= TED     TRAILER+ TIONS- TIONS   GES *MASTER-WRITTEN* DIFF.  COUNT
------  ------------------   ------ ------ ------  ------- ------ ------ ------ ------ ------- ------ -------
  01    INVALID RECORD CODES
  02    COMPANY NAME TABLE                               2                           2       2
  04    RESPON. NAME TABLE
  06    ORGAN. NAME TABLE
  10    LD-STAT. OF OPER
  14    LD-ACCOUNTABILITY RT
  18    LD-BUDGET COMPARISON
  22    LD-BALANCE SHEET
  26    CONTRACT TYPE DESC.
  30    FUNDING SOURCE DESC.
  34    CONTRACT STATUS DES.
  38    PAPER STOCK DESC.
  42    VALIDATION PARAMETER
  46    CONTRACT DESC. TABLE
  48    SOURCE CODE DESC.TBL
  50    ACCOUNT DESC. TABLE
  54    ORGAN. CODE VAL. TBL
  58    REPORT DESC. TABLE
                             ------ ------ ------  ------ ------ ------ ------ ------  ------ ------ ------
  00                                                     2                           2       2
                             ****** ****** ******  ****** ****** ****** ****** ******  ****** ****** ******
```

Fig. 3–19 Example of output from a conventional line printer

The most common impact printer is the chain printer. In this printer, the characters are in a circular steel chain which moves horizontally in front of the paper. Hammers in back of the paper press the paper forward against the type-characters on the chain, thus impressing the character onto the paper. Only one revolution of the chain is required to print a line. If an "E" is to be printed 28 times in one line, the "E" character on the chain will be printed each time it passes in front of a print position on that line where an "E" is required.

Another common impact printer uses a metal drum engraved with rows of characters. The drum revolves during printing, moving each row of characters past the print hammers. As the proper characters pass by, the hammers press the paper against the drum to print.

A type-wheel printer is another impact printer, with each printing character on an individual metal print wheel. Still another impact printer forms each character as a pattern of dots inprinted by the ends of small wires. As you can see, each of these printers works in essentially the same way.

Nonimpact printers operate with chemically impregnated paper which is energized by pulses of current or dusted with a deposit of electrostatically charged ink. Nonimpact printing techniques are available which will print as fast as 31,250 lines per minute. But you can

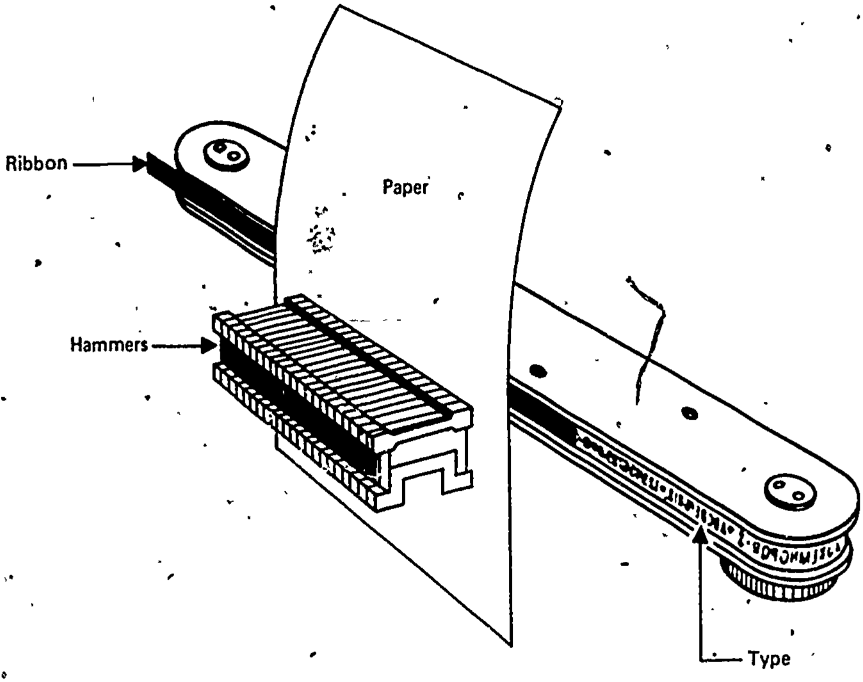


**Fig. 3-20** The chain printer mechanism

imagine how expensive such a printer would be! (Do you think the cost would be justified by the extremely high speed in some cases—for example, for certain kinds of output in a space flight project?)

Line printers are usually quite versatile and can be adjusted to print on various sizes and weights of paper. They are commonly used, for example, to print reports, checks, report cards, invoices, and just about any other type of output that can be printed on paper or forms.

Since it is most common for computer systems to use a printer for a major portion of its output, the computer operator usually becomes quickly familiar with loading and adjusting the printer in his or her system.

## Card punch

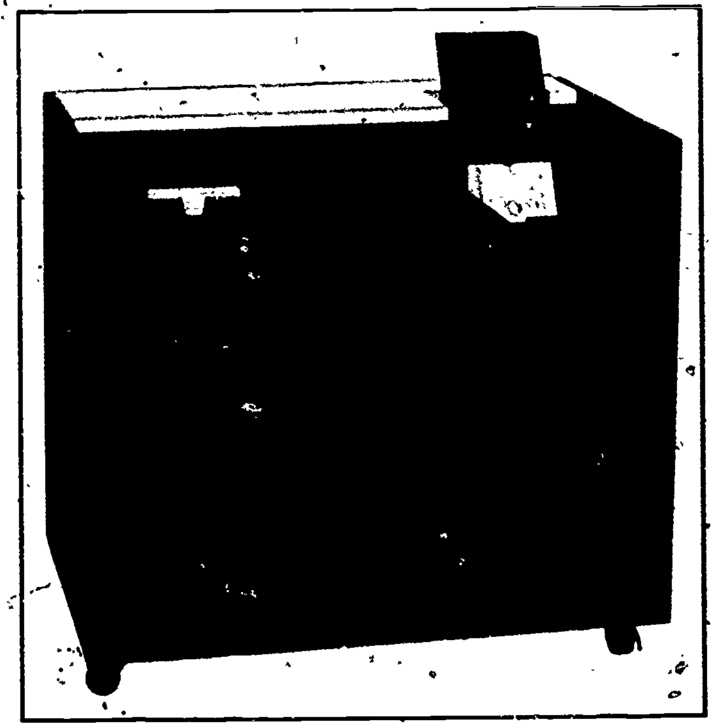
There are times when the output from a computer is data which need to be stored on punched cards. An example would be when the data will be used at a later time as input data.

You are already familiar with the medium of punched cards, which you read about on pages 69 through 77. There you also learned about the keypunch machine which the human operator can use to punch data onto cards for input. As you saw, the cards are input by a card reader connected to the computer. But, how can a computer get its output punched onto cards for future use?

The device which the computer uses to output information on punched cards is called, naturally enough, a card punch. It is always connected directly to the computer. Frequently the card punch is combined with a card reader, as is shown in the illustration below. When the two devices are combined, the unit is usually called the card read/punch.

Whenever a computer run calls for output to be produced in the form of punched cards, the operator is responsible for loading the card punch hopper with blank cards and for readying the device for use when the computer calls on it. When the computer activates the device, cards are automatically drawn, one at a time, from the hopper through the punch station where the data is punched in coded holes into the card. Once punched, the card is deposited in the unit's stacker.

**Fig. 3–21** A typical card read/punch



Since they are punched and deposited one at a time, the cards are always kept in order. When the run is through, the operator has only to remove the complete card deck and store it or route it to the programmer, depending on what instructions he or she has received for the deck.

### Paper tape punch

You will remember from the discussion on pages 81 through 85 that paper tapes can be punched by a machine operator using a separate paper tape punch or a teletypewriter with a paper tape punch attach-

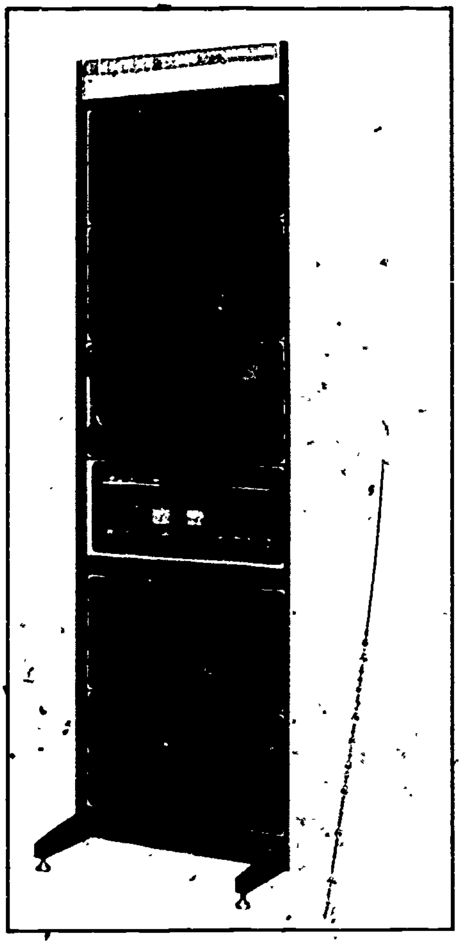


**Fig. 3–22** A paper tape reader punch

ment. Both of these devices are equipped with keyboards on which the operator may type the data to be punched.

When the computer needs to output data in the form of paper tape, it most often uses the output device called a paper tape punch, which is connected directly to the computer. Once the computer operator has loaded the paper tape punch with a roll of blank tape, the computer can activate the punch at any time. There is, of course, no keyboard on this device. The computer controls the punch mechanism entirely electronically. It sends impulses to the punch station to punch the proper sequence of holes to represent each piece of data. When all output has been produced on the paper tape for any given run, the operator removes the coded tape and stores or routes it according to the instructions for the run.

If you will go back and reread page 105, you will see that the computer can also use the paper tape punch attached to a teletypewriter as a paper tape output device. The teletypewriter need only be connected by phone to the computer and the punch turned on for the computer to be able to use it for output purposes. But, remember, this is a remote terminal usually used at a distance from the computer.

## Magnetic output devices

The magnetic tape, disk, or drum devices you studied on pages 105 to 113 as input devices can serve as output units as well. Whenever a program requires output onto a magnetic medium, the computer electronically activates the read-write assembly of the appropriate device and transmits all data to it to be "written" (encoded in magnetic spots) on the medium. Any information output onto a magnetic tape, disk, or drum can be easily stored in the computer room for future use.

When a computer program specifies a magnetic medium, say a disk, for output, the computer operator loads the appropriate disk or disk pack onto the disk drive. When the computer comes to the part in the program requiring this output means, it picks up the disk and track location specified by the program and transmits that location to the disk drive's read-write head. The read-write head is activated, and it reads until it reaches the desired location. Then the computer transmits the data that are to be written on the disk to that read-write head, which records the data onto the disk.

Once all the information has been output onto the magnetic disk or disk pack, the medium is stored for future use.

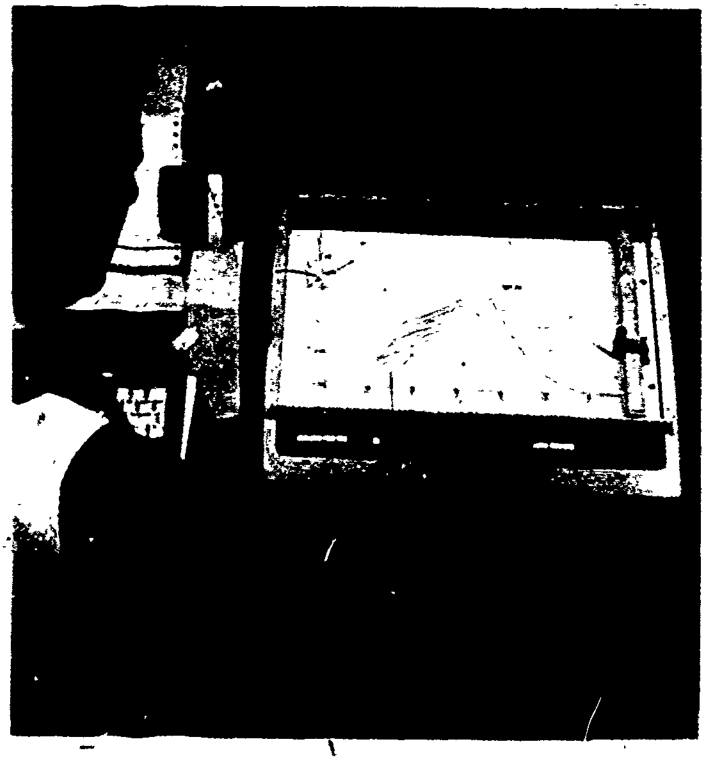


Fig. 3–23 A plotter drawing a graph

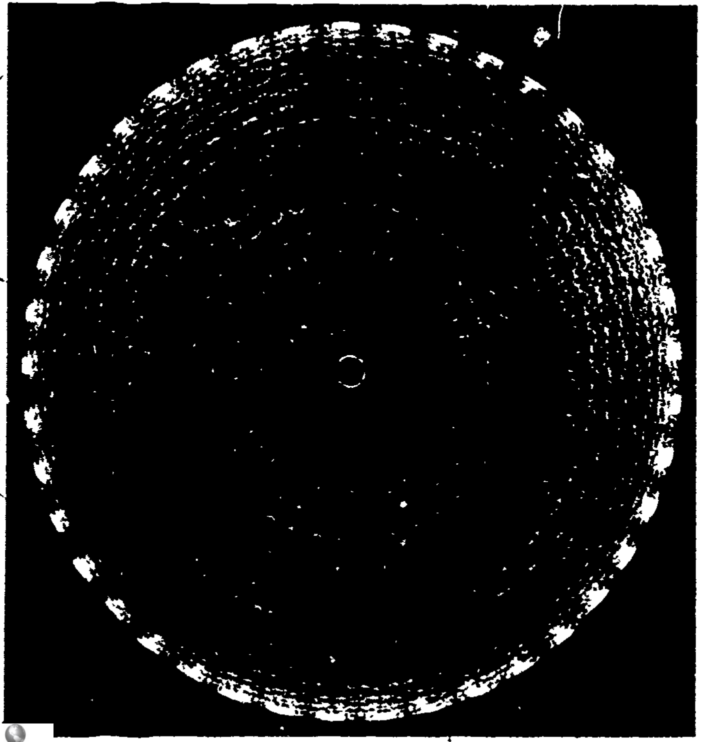


Fig. 3–24 A plotter design

### Other output devices

In addition to printed words, punched holes, or magnetized spots there are other forms of computer output such as graphics and even voice output.

One fairly common graphics device is called a "plotter." A plotter has a movable pen controlled by the computer to draw graphs or designs, as shown in Figures 3-23 and 3-24.

Graphs and creative designs can also be output in picture form on terminals with TV-like display screens. Engineers commonly use such terminals to help them design and modify drawings. The drawings can be stored in the computer and displayed on the screen so the engineer can "draw" chan‹es with the light pen.

See how well you understand output devices by answering the following questions.

## Check your understanding

1. The difference between "data" and "information" is

   a. data are "unprocessed information."
   b. information is input, data are output.
   c. data are usually not meaningful to human beings; information is meaningful.
   d. both a and b.
   e. both a and c.
   f. all of the above.

2. The most common output device in most computer centers is

   a. the line printer.
   b. the teletypewriter.
   c. the card reader.
   d. the CRT.

3. Line printers

   a. print about 100 characters per line, one character at a time.
   b. ca, be activated by an input device.
   c. print an entire line at a time.
   d. can print on only one size paper.

4. A card punch

   a. is another name for a keypunch machine.
   b. is frequently combined with a card reader.
   c. is activated by a card punch operator.
   d. can be used as an input device.

5. A paper tape punch used as an output device

   a. is activated by the computer.
   b. may not have a keyboard.
   c. may be a part of a teletypewriter.
   d. all of the above.

6. Which of the devices below is (or can be used as) an output device:

   a. Card read/punch
   b. Paper tape punch
   c. Teletypewriter
   d. Magnetic tape drive
   e. Line printer
   f. Plotter
   g. Magnetic disk unit
   h. Magnetic drum unit
   i. A display screen on a terminal

## TERMINALS: REMOTE INPUT/OUTPUT DEVICES

As you probably are aware, in the present day more and more companies and government agencies have their operations dispersed or decentralized into branch offices around the country. As organizations find it increasingly essential for data to be collected, processed, and analyzed without delay, more and more use is being made of remote terminals. Through these, distant branch offices can send their data to a central office in a matter of minutes for processing which, in turn, may take only a few seconds.

Why don't small businesses and branch offices of larger companies just install their own computers? Many organizations find it simpler, more convenient, and less expensive to lease a terminal which can be connected to a central computer than to install their own computer system. The demand for remote computer access has encouraged the development and speed of time-sharing computers and terminals.

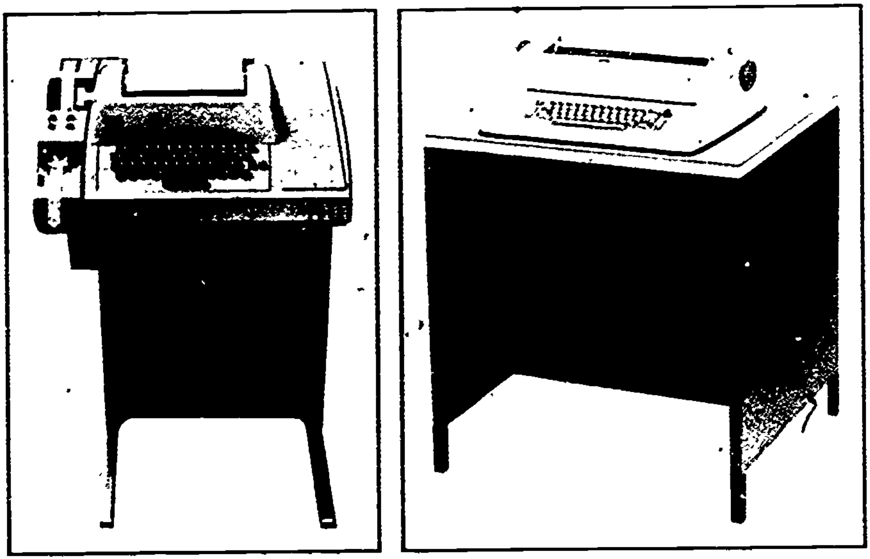
You will remember in our earlier discussion of time-sharing that we talked about "terminals" as input and/or output devices located at a distance from the central computer system and connected to the computer by telephone lines. A "terminal" can be as simple as an electric typewriter, or it can be much more complex, with additional input or output devices and attachments. Let's look at the simplest kind of terminal first, then go on to consider terminals that have more and more capability and complexity.

### Keyboard terminals

You have probably already used one of the simplest and most common keyboard terminals available—the teletypewriter. To use it, the data clerk simply makes a telephone connection to a distant computer, types a coded "password," then types all input at the keyboard. Everything the user types is converted to pulses and transmitted over the telephone line to the computer center.

There are many other keyboard terminals in use, all of them similar to the teletypewriter. The IBM 2741, for example, is commonly used as a terminal in an IBM computer system.

**Fig. 3–25** Some keyboard terminals. Teletype (left) and IBM 2741 (right)



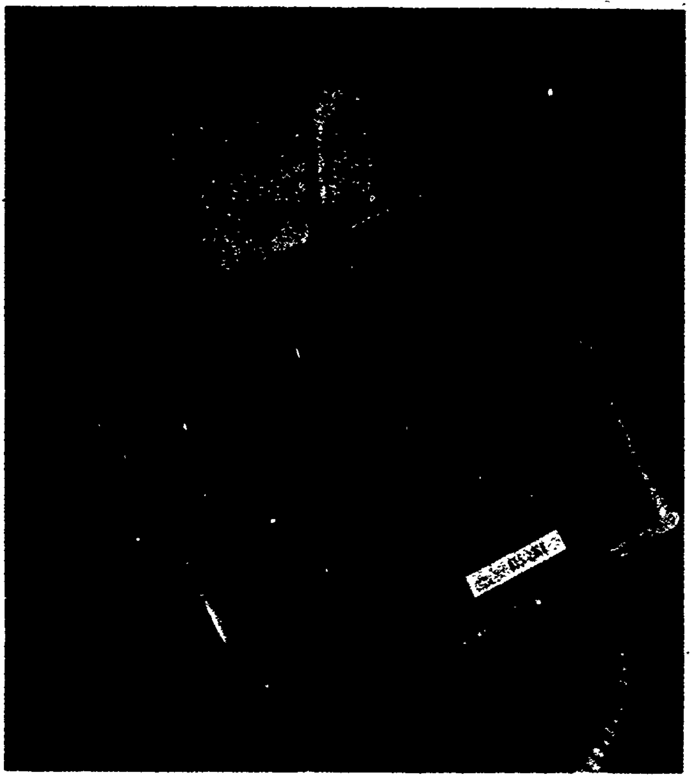


Fig. 3–26 A card dialer telephone terminal

Most keyboard terminals can be used "off-line"—that is, not connected to the computer—for ordinary typing.

An even simpler "keyboard" terminal is one you are very familiar with. It doesn't have a printer, and transmits only numbers. It is the telephone itself. One type of touch-tone telephone, for example, is designed so you can automatically dial a computer by inserting a plastic card with holes prepunched in it. Then you can insert a gasoline credit card with prepunched holes for the account number, and finally enter the amount of purchase by touching the correct number keys on the telephone. This is called a Card Dialer.

### Punched paper tape terminals

Most terminals which transmit and receive punched paper tape are the familiar teletypewriters with paper tape attachment. You will recall that the attachment for transmitting input data is the *paper tape reader*, and the attachment for receiving output data from the computer is the *paper tape punch*.

Do you recall another use for the tape punch besides receiving output data from the computer? Of course! It is used "off-line" to prepare the punched paper tape for later input to the computer.

### Cathode ray tube terminals

Some terminals, instead of printing data as it is typed and transmitted, display the data on a TV-like screen called a *cathode ray tube* or CRT. Can you see advantages and disadvantages in using a visual display on a screen rather than a typed copy? One advantage is that a CRT can usually display information much faster than a typewriter can type it, and even faster than you can read it—which may be a disadvantage.

Another disadvantage in some cases is the lack of "hard copy," that is, a printout to keep for future reference. A CRT is very quiet as compared to a typewriter, so is more useful in a busy office environment. Look at the examples of CRT's in Fig. 3-27.

### Card reader terminals

Just as data from paper tape can be transmitted to a computer through a paper tape reader, so can some punched (or marked) card readers be used as remote terminals for reading and transmitting data. The cards are punched on a keypunch (or marked with a pencil), then read in through the card reader, which is connected to a telephone line. Fig. 3-28 shows two card reader terminals.

Use of a terminal which will read either punches or marks on a card can simplify many operations. For example, meter readers for the electric company might receive a deck of cards for all of their accounts, each prepunched with the customer's name, address, and account number. When they read the meters, they simply mark the amount with a pencil, on the same card. At the end of the day, these punched-and-marked cards are read in to the computer at the electric companys headquarters, and the customers' accounts are updated for billing.

Likewise, the computer can transmit output to a card punch terminal to be punched into cards. These punched cards may later serve as input to a computer.

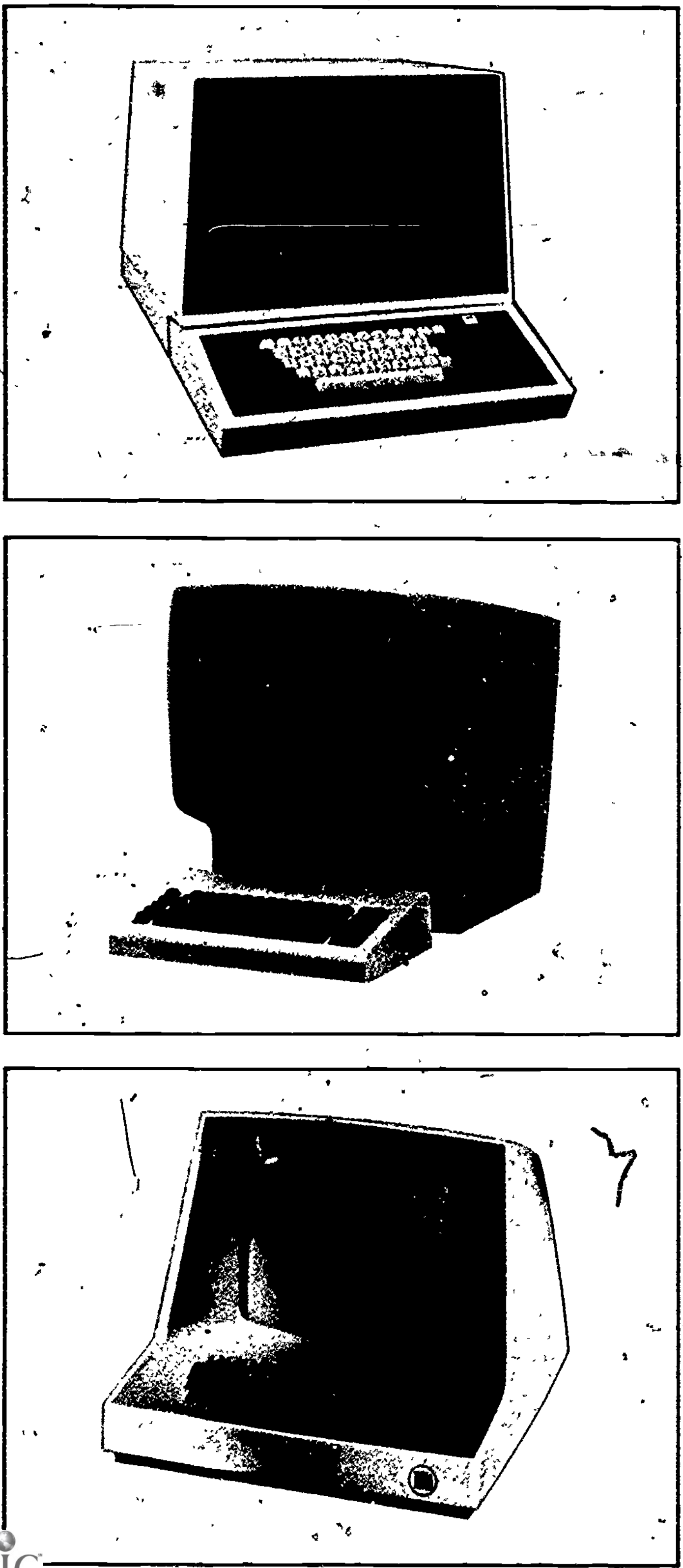


Fig. 3–27 Sample CRT's. Hewlett-Packard 2615A (top), IBM 3270 (middle), Lear Siegler CRT terminal (bottom)

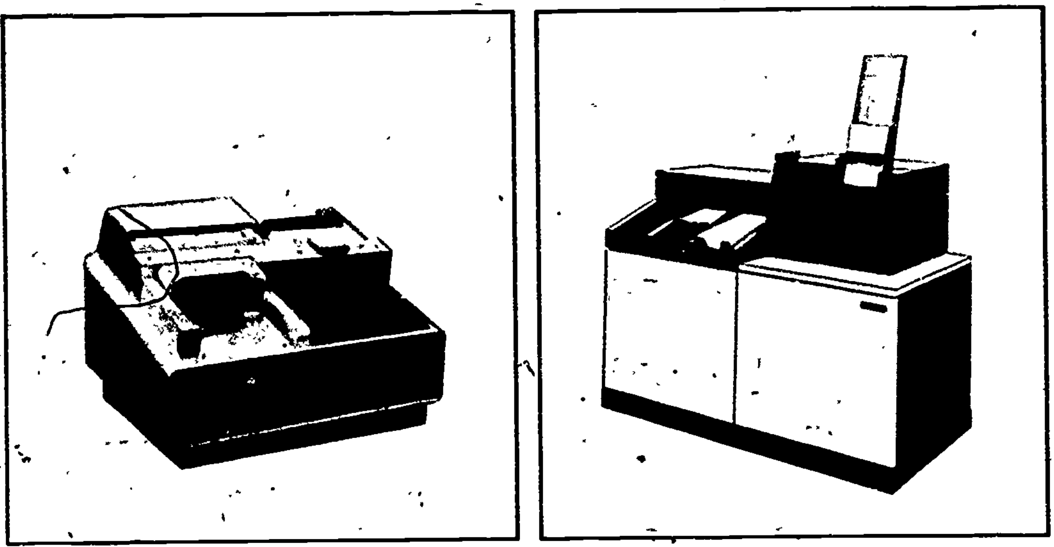
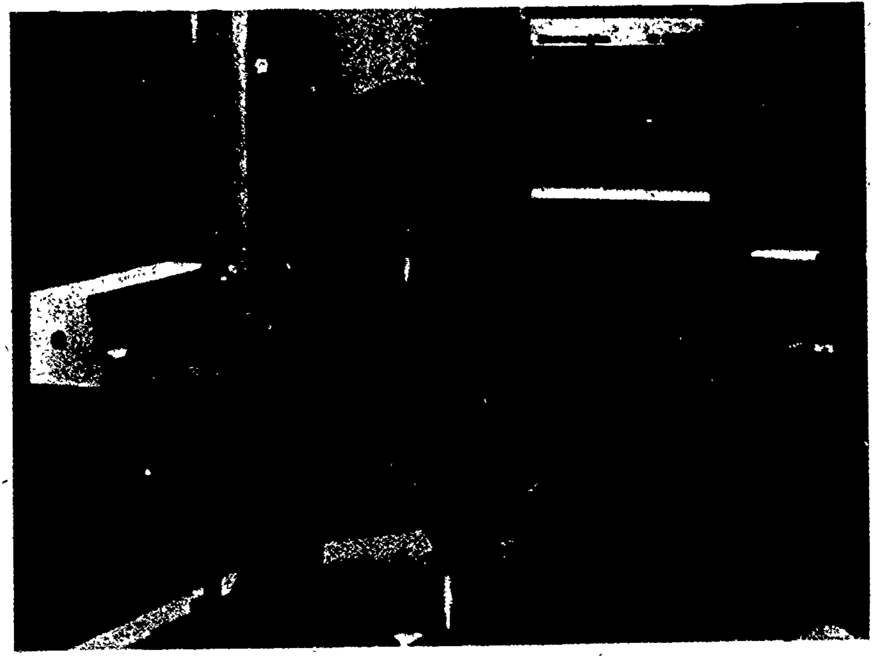


**Fig. 3–28** Typical card reader terminals. Hewlett-Packard 2893A cārd reader (left), IBM 3505 card reader (right)

**Fig. 3–29** Card punch terminal with Burroughs B 1700 system



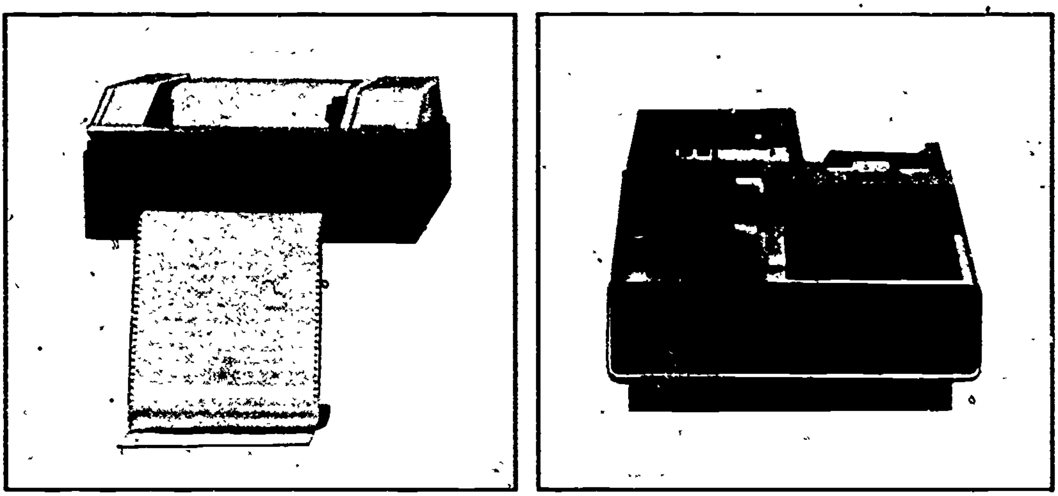


Fig. 3–30 A Hewlett-Packard line printer together with a card reader used for input/output

### Line printer terminals

Sometimes a remote terminal may include a line printer to rapidlv print computer output an entire line at a time. If this is the only device present, then the terminal is obviously limited to receiving output transmitted from the computer. However, most such terminals include an input device as well—perhaps a card reader or keyboard.

### Complex terminals

Various input or output devices can be added to a terminal site until it consists of several different pieces of equipment. In a single location there may be a teletypewriter with paper tape punch and reader, a punched-or-marked card reader, a card punch, and a line printer. For example, look at the arrangement of terminal equipment in Fig. 3-31. How many different input/output modes can you count in this example? You should be able to identify four input modes (Tty keyboard, CRT keyboard, tape reader, and punched/marked card reader) and five output modes (Tty keyboard, tape punch, card punch, printer, and CRT display screen). Or another arrangement might include a CRT, a punched card reader, and a line printer, as in Fig. 3-32.

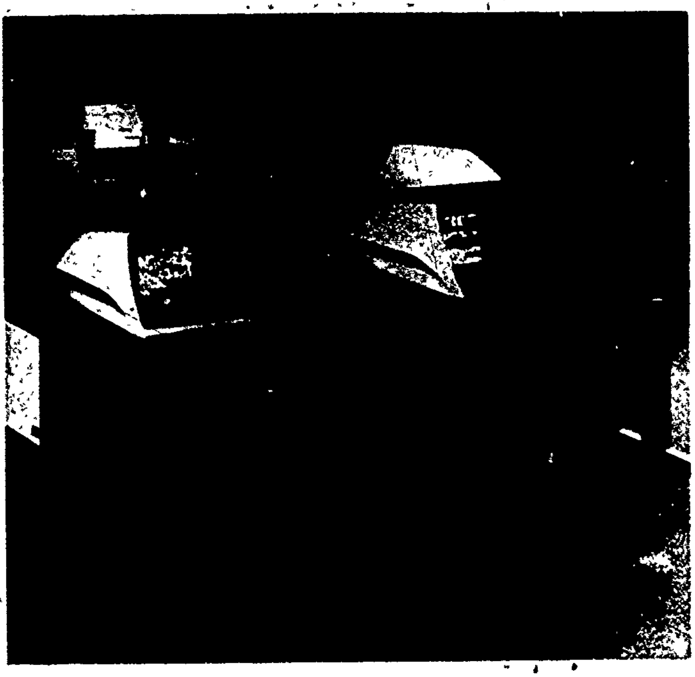


**Fig. 3–31** Arrangement of input/output devices

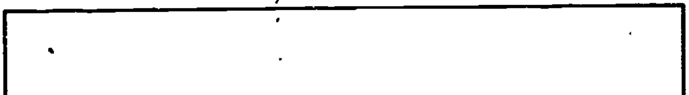


**Fig. 3–32** A CRT, punched card reader, and line printer

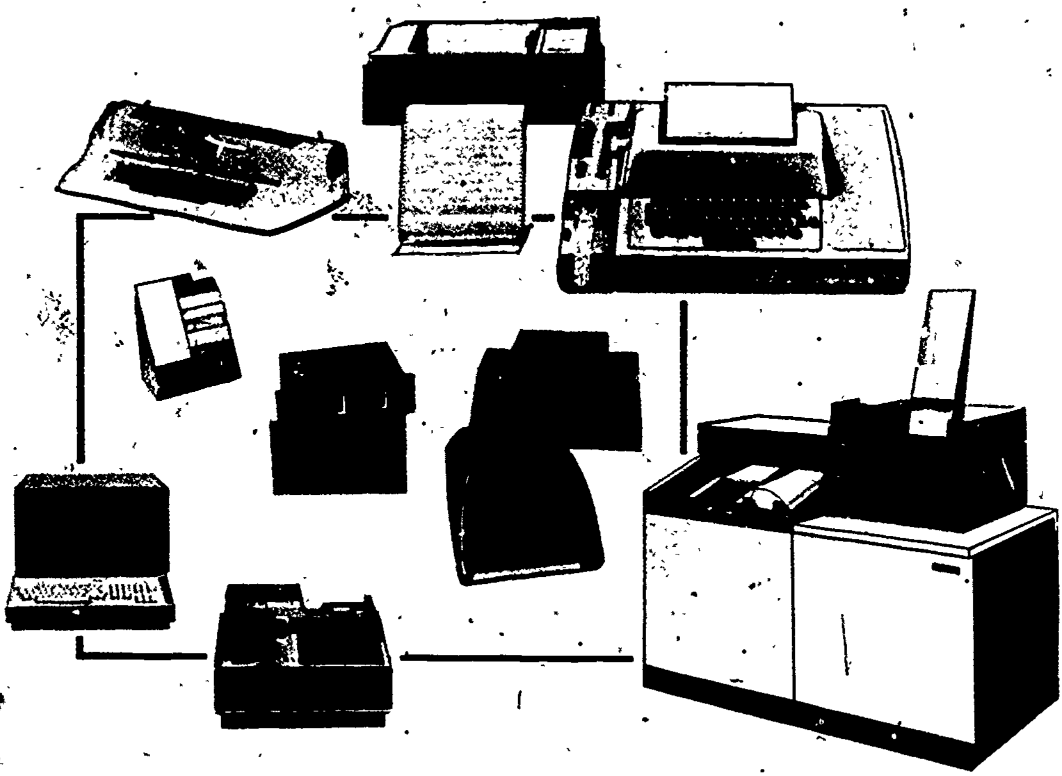


Fig. 3-33 An array of input/output devices

S.

There is virtually no limit to the number of possible combinations of input/output devices for terminals. Look at the array pictured in Fig. 3-33.

When more than one or two input/output devices are combined in this way, it quick'y becomes advisable to add a processing unit with a small amount of storage in order to *control* the various input and output operations. This might be diagrammed as shown in Fig. 3-34.

Does this look familiar? It should. Do you recall a type of terminal we have discussed before, with a mini-CPU of its very own, plus a varying number of input/output (and even storage) devices? Look at the pictures of this type of terminal in Fig. 3-35. Can you recall what this kind of remote terminal, with its own mini-processor, is called? (Look back to pages 55 and 56 if you need a reminder.)

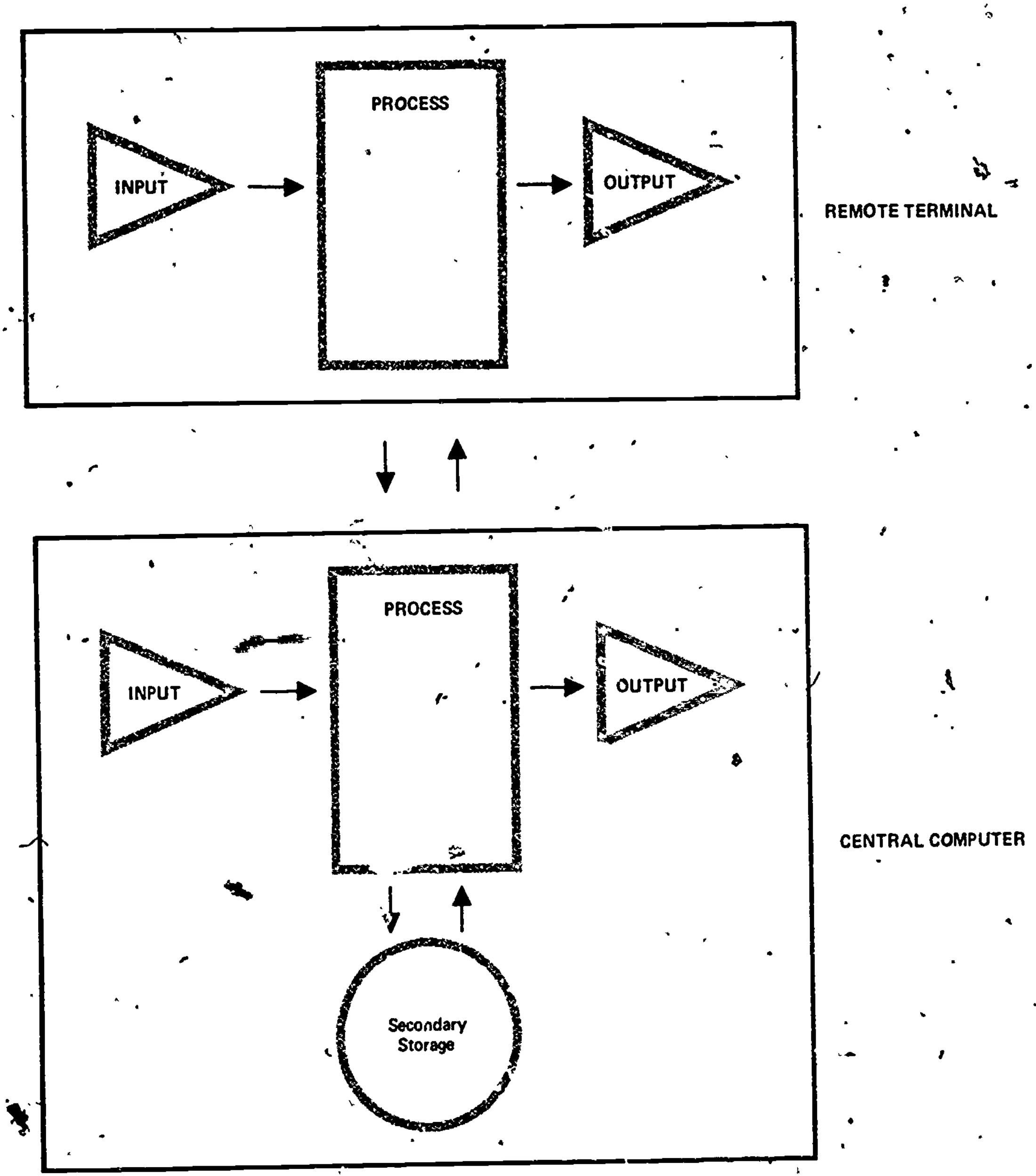


**Fig. 3–34** Illustrat:on of remote terminal with processor connected to a centra' computer

Fig. 3–35 Intelligent terminal with storage device

## Interesting devices

"Remote terminals" don't have to be keyboards or readers or printers. There are dozens of "terminals" that do transmit data to a central computer, but don't require a trained operator.

One interesting example is the automatic reader that helps keep track of railroad freight cars. Have you ever noticed (and wondered about) a patch on the side of a freight car, with many different colored bands? These colored bands are put together in a coded combination to identify a particular car. When the car passes a certain point in a freight yard, an Automatic Car Identification Scanner automatically reads the colored bands and transmits the code to an output device or a central computer, depending on the way the scanner information is to be used.

Another interesting example is the CRT equipped with a "light pen" as the input device. A light pen is a pen-sized tube with an electronic device in it and a light bulb at its tip. The user holds the light pen in his or her hand and points it at the screen. The location of the place pointed to is transmitted to the computer. A frequent use of this type of terminal is in computer-assisted instruction, where the computer might, for instance, display a word along with several pictures and ask the student to pick the picture that matches the word. The student points the light pen at the correct answer, and this is transmitted to the computer, which then flashes a complimentary message on the screen. (Or the student points at the wrong answer, and the computer displays some other message, or gives a hint and lets the student try again.)

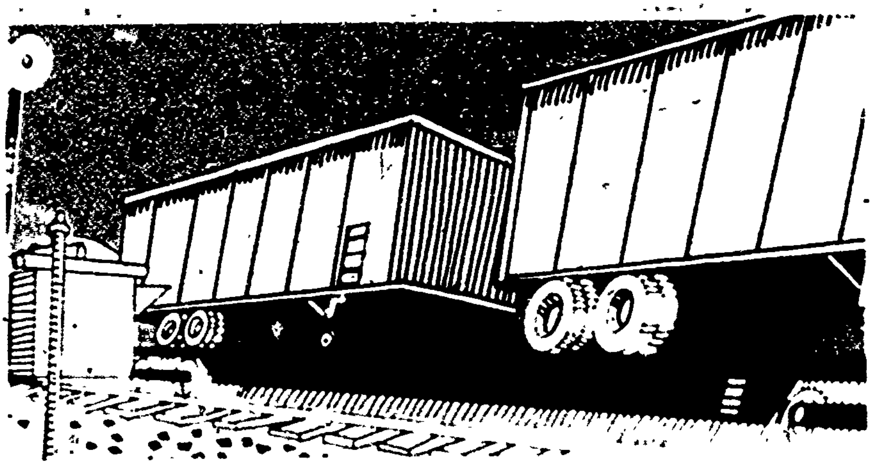


Fig. 3-36 Railroad cars marked with coded color-bands for identification by Automatic Car Identification Scanners

Fig. 3-37 Students may use light pens to indicate answers on a CRT screen.

Another hand-held electronic device for transmitting information is an electronic "wand" which can read magnetically coded strips. Millie Fogarty, in Chapter One, saw such a device in use during her shopping errands. The wand is used to read the information coded on the tickets of purchases, such as price, color, stock number, and clerk's department number. This is then transmitted to the store's central computer. If the customer presents a credit card, the electronic wand reads the account number from a magnetically coded strip on the card. This information, too, goes immediately to the central computer for a credit check. That night, all the day's data are transmitted from the store's computer to a regional data center for processing.

An exciting development in input/output is voice input and voice response. Once perfected, voice input/output will obviously be a superior way of communicating with a computer. No data preparation is necessary, and errors in entering data are rare. Output is speedy and easily understood. In a restricted security area, persons entering the area could speak into a terminal, which would transmit the speaker's voice pattern to a centı. l computer to be compared with a stored bank of "authorized" voice patterns. Or a credit card could have a voice pattern prerecorded on it for identification and comparison.

Analyzing and recognizing voice patterns is not difficult for a computer, as each number and each special word has a distinct pattern different from other numbers and words. (As you might imagine, however, a Southern or British accent could confuse a computer programmed in Minnesota!) Also, each human voice has a unique recognizable pattern unlike any other human voice, just as fingerprints have patterns unique to an individual.

Getting the computer to "talk back" is a different problem. The computer needs access to a vocabulary bank of prerecorded spoken words and numbers which it can select from to put together replies. A bank teller, for example, speaks into a special terminal to get information about a customer's account—say, account number 8106. The computer locates the account and answers in an audible voice over a loudspeaker or a telephone receiver. The spoken output message might be "Account number eight one oh six balance to date three seven two point five nine." (Account number 8106, balance to date $372.59.)

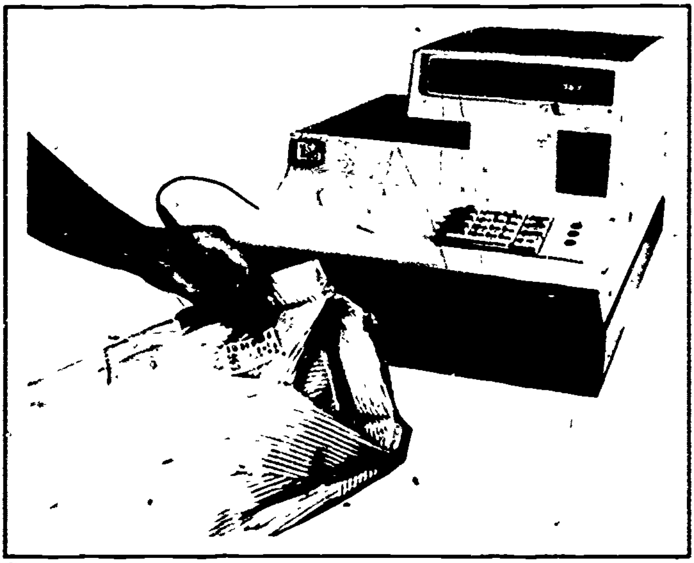


**Fig. 3–38** Electronic wands can read from magnetically coded strips.

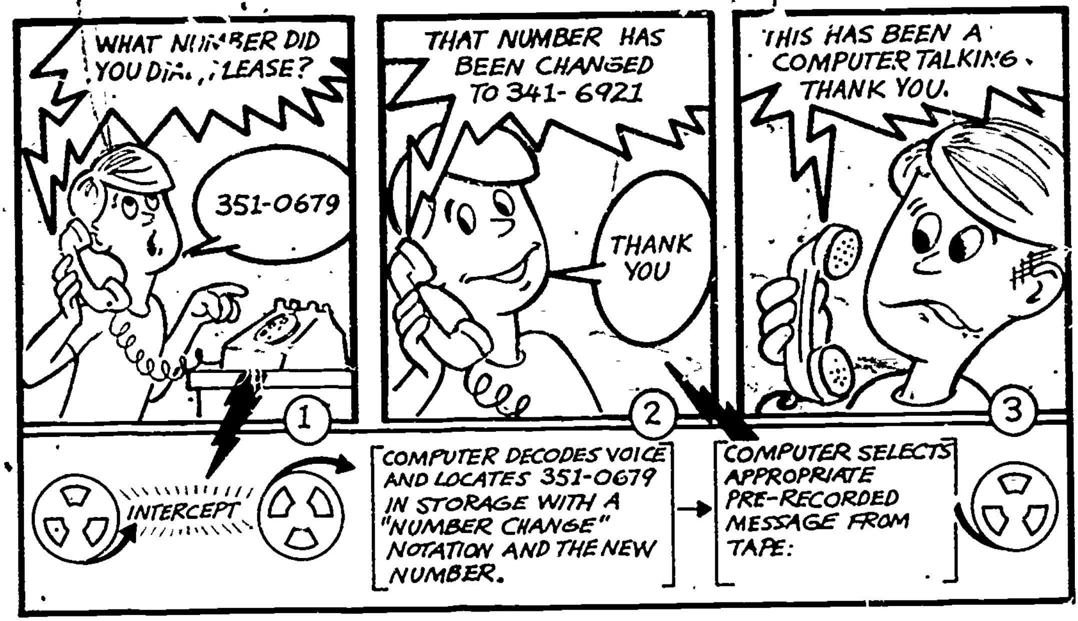

WHAT NUMBER DID YOU DIAL, PLEASE?
351-0679
THAT NUMBER HAS BEEN CHANGED TO 341-6921
THANK YOU
THIS HAS BEEN A COMPUTER TALKING. THANK YOU.
1
2
3
INTERCEPT
COMPUTER DECODES VOICE AND LOCATES 351-0679 IN STORAGE WITH A "NUMBER CHANGE" NOTATION AND THE NEW NUMBER.
COMPUTER SELECTS APPROPRIATE PRE-RECORDED MESSAGE FROM TAPE:

Fig. 3–39 Obtaining a changed telephone number from a computer

Perhaps you have had the uncomfortable experience of dialing an out-of-service number and having the call intercepted by a voice asking for the number you dialed. When you repeat the number, the voice replies that the number is out of service, or has been changed, and may even give you the new, correct number. You thank the voice, which then informs you that you have been talking to a computer!

Can you picture the sequence of events?

Can you think of other types of terminals you have seen or used? Perhaps you have seen special "ticker tape" machines in a stockbroker's office, which do nothing but output stock transactions and prices on a narrow tape, hour after hour. Or a remote "plotter" terminal which draws pen and ink graphs or designs under computer control. There seems to be no end to the ways of transmitting data from humans to computers and back to humans.

### Ways terminals are used

We talked earlier about "conversational," or *interactive* use of the computer from a terminal. Do you recall the characteristics of interactive use? Remember, as soon as a user enters some input, the computer responds with output, then the user enters some more input, and so on. Who do you think would be most likely to use an interactive system? Students, obviously, might write and store a program to do math problems, or one that calculates data from a science laboratory. They might use the computer to practice arithmetic skills, help them learn a foreign language, or explore the ways a political system operates.

An engineer could use an interactive terminal to do calculations, retrieve data, or try various formulas with various sets of data. Through an interactive terminal, a company could handle its bookkeeping, billing, payroll, and inventory control. A researcher in a laboratory might find an interactive terminal as vital a tool as any research instruments or equipment.

Earlier we talked about "batch processing," where the computer center collects jobs to be run from several users, then enters all the jobs in a single "batch" separated by control cards.

It is possible to do batch processing from a terminal, simply by collecting several jobs before dialing up the computer, entering all the jobs at once, and, at a later time, receiving all the output at once. This kind of batch processing done from a terminal is called "remote batch," for obvious reasons. Some computer centers call it "quick-batch," if the output is transmitted back to the terminal within a few minutes.

When data are entered from a terminal it's usually done in remote batch mode. Data are first collected at the site of the terminal until there are enough to transmit in a single batch. Then the operator uses the terminal to enter all the data as a single batch. The terminal may be connected directly to the distant computer (called *on-line* transmission), or the data may be transmitted from the terminal to an *off-line* device, such as a magnetic tape unit, to be stored for processing by the computer at a later time.

In an automated version of remote batch data entry, the central computer signals the terminal at a designated time—when it is ready to receive the data, say, midnight—and the terminal automatically turns on and begins to input data from previously prepared punched tape, punched cards, or magnetic tape. In this way a data clerk at a branch office can prepare a data tape at the end of each business day, place it on the tape reader of the terminal, and then leave. During the night the central computer at headquarters would follow a schedule to signal the terminal in each branch office, receive all the day's data, and process it before the next business day started.

You may have heard of terminals being used for inquiry This is the mode used by airlines and hotel chains for their reservation systems.

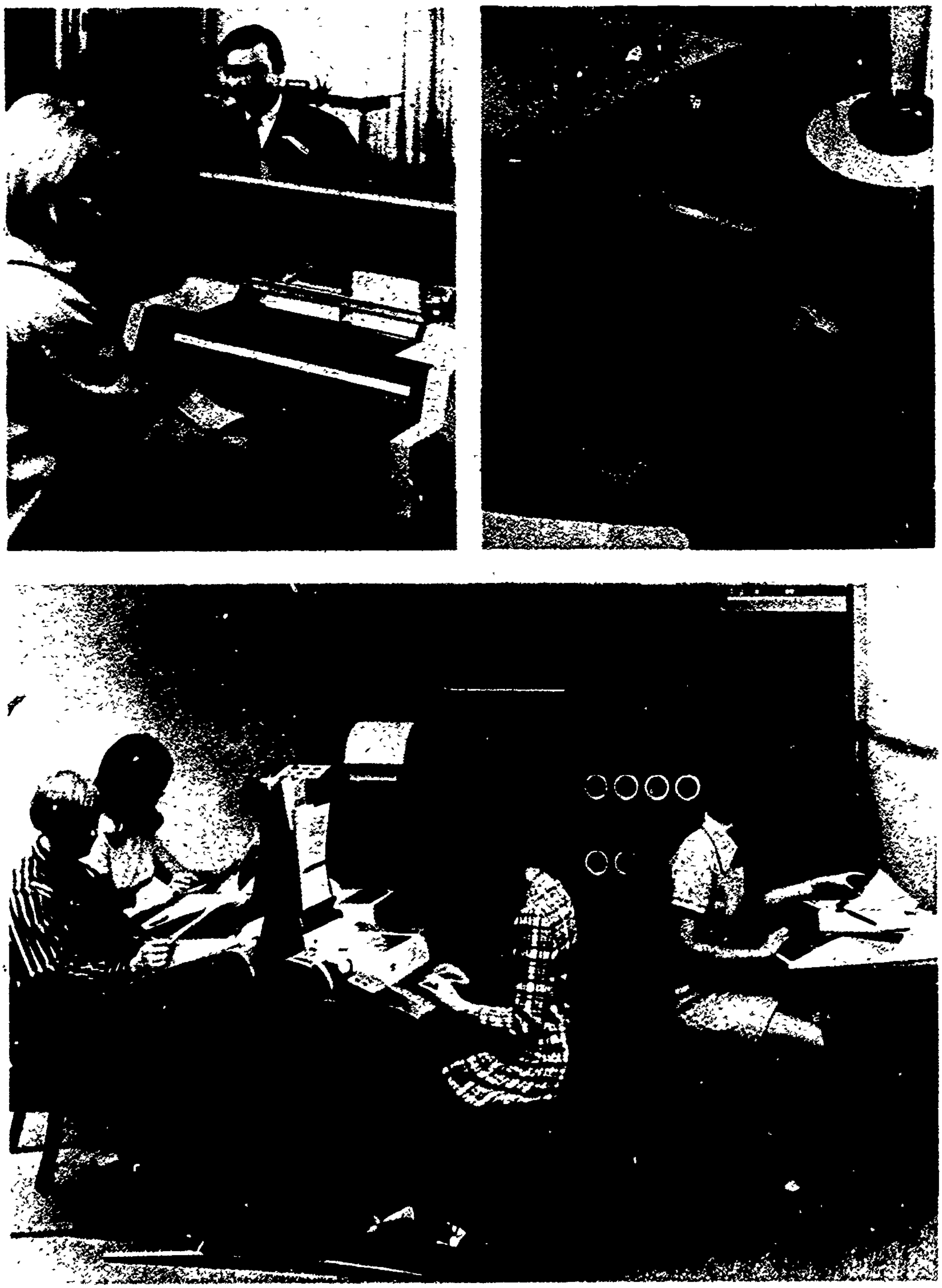


Fig. 3-40 Many people use terminals for many things.

**Fig. 3–41** Airlines make extensive use of inquiry mode, using CRT terminals.

The remote terminals are linked to central data storage files, and a clerk in a hotel uses the terminal to question the data files to see if the hotel has a vacancy available for a given date. Each new reservation or cancellation updates the stored information to show that reservation (and one less vacancy).

In other instances, a terminal may be used for simple inquiry, or information retrieval, without changing or updating the stored information. An example of this would be in an insurance office, when the agent inquires about the status and coverage on a customer's car insurance. The information might be displayed on a screen, but the agent would not change it in any way.

Some terminals are used exclusively for remote printing, with no input device available. The teletypewriters in a newspaper's news room, constantly printing stories from UPI or AP news services, are remote printing terminals. So are the stockbroker's ticker tape terminals. Weather bureaus usually have a terminal that does nothing but print weather forecasts and conditions from a central agency.

## Other kinds of lines for connecting terminals

So far, we have only mentioned telephone lines as the means for connecting a terminal to a computer. This is, indeed, the most popular mode of transmission. However, a company may install its own direct communication lines, permanently connected to the terminals for direct service without having to dial a telephone. Microwave transmission may also be used instead of telephone lines.

Check your understanding of terminals, now, before going on to study the computer's central processing unit.

## Check your understanding

1. Which of the following are types of keyboard terminals?

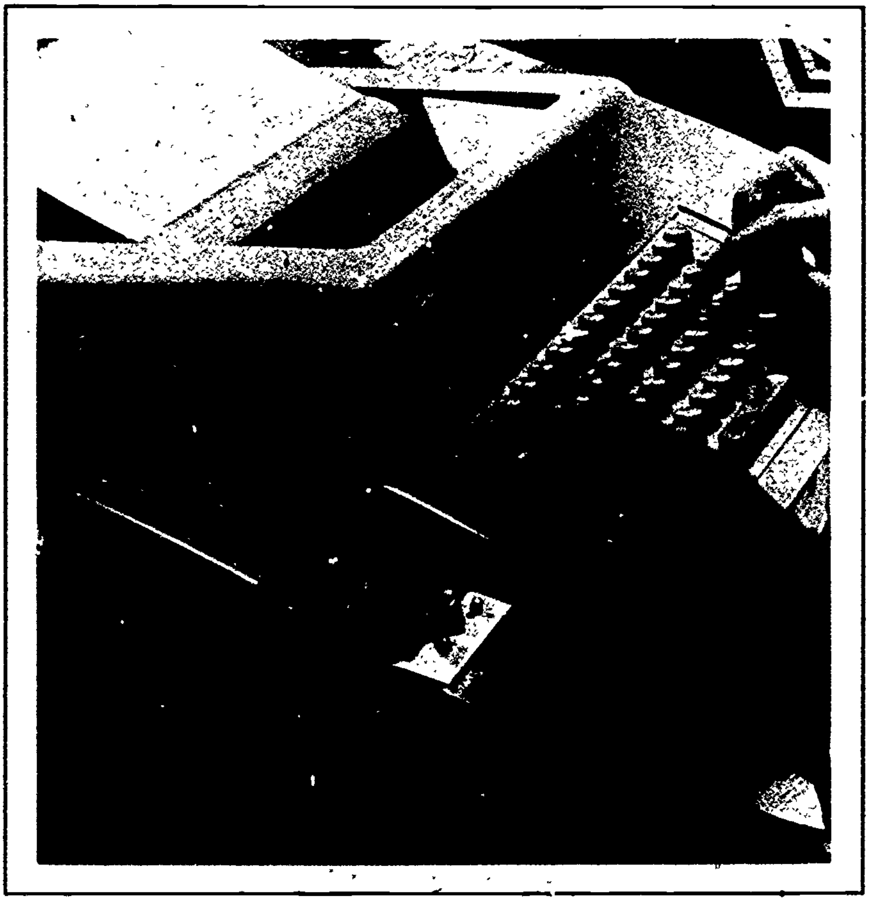
   a. Adding machine
   b. Teletypewriter
   c. Telephone
   d. Line printer
   e. CRT

2. What is a cathode ray tube terminal?
3. Which of the following is the most complex combination of input/output devices possible for one remote terminal?

   a. Line printer, card reader, CRT, and CPU
   b. Teletype, line printer, card reader, CRT, and CPU
   c. Paper tape reader, teletype line printer, card reader, CRT, storage devices, and CPU
   d. There is no practical limit to the number and combinations of input/output devices at one terminal.

4. What is the terminal called which is composed of one or more input devices, one or more output devices, and a processor of its own?
5. What are some practical uses for a remote terminal?
6. Which mode (batch or interactive) is used by terminals at an airlines reservation desk?
7. When data are collected for a period of time, then entered all at once from a terminal, what is it called?
8. Describe some uses for terminals which have output capability only.
9. Does your school have a remote terminal? If so, describe which devices it includes.

123**4**567



# The computer itself

## CENTRAL PROCESSING UNIT

Now that you are familiar with the input and output devices most often used in computer systems, you can spend some time learning about the heart of the system: the central processing unit (or CPU). It is here that the input data is processed to result in new output information. In most computers, the CPU is housed in a single cabinet or console,

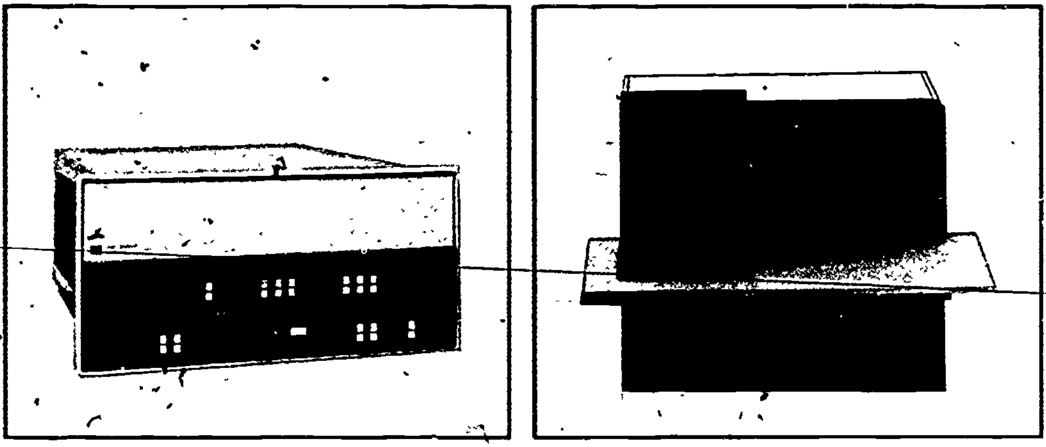


Fig. 4–1 Central processing units vary in size and capability. Hewlett-Packard mini-computer (left) and IBM 370 (right)

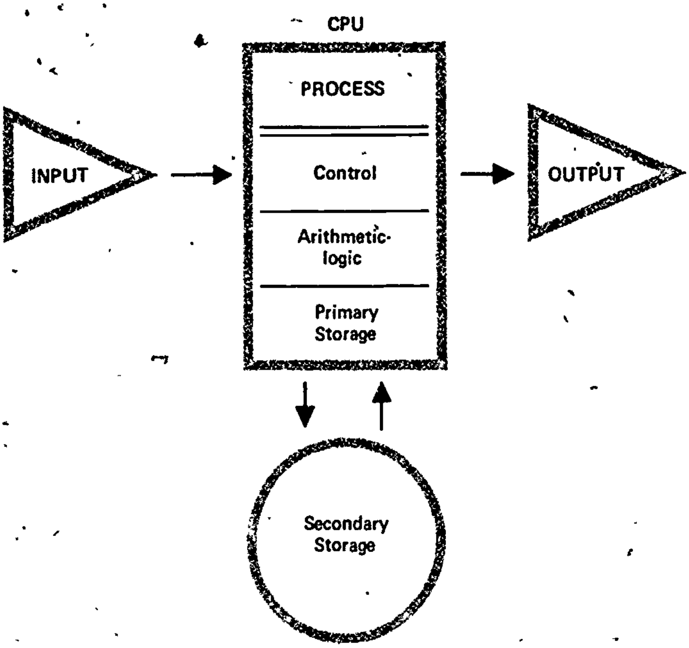


Fig. 4–2 The three parts of the CPU: control unit, arithmetic/logic unit, and primary storage unit

sometimes called the "mainframe." Although the central processing unit appears to be a single piece of equipment, it actually has three distinct parts. One stores the programs and data for use in the run (the storage unit), one takes care of the actual processing or manipulating of data (the arithmetic/logic unit), and one coordinates the flow of data through the system (the control unit). We will look at each of these parts separately now, starting with the primary storage unit.

## Primary storage unit

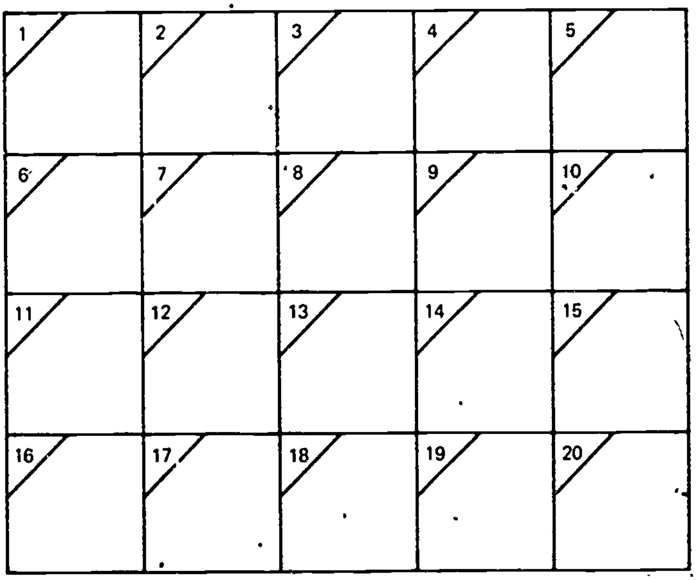
You have already seen that a computer system must have both a program and data in it before it can process the data or produce results. These are usually loaded into the computer through input devices or from secondary storage devices such as magnetic tape or disk just before the particular run is made. Normally, the program is loaded in first and then the data to be processed are input.

Did you wonder where the program and data went once "loaded" in? Here's your answer. All input is loaded directly into the CPU's storage unit, usually called "primary storage" (or "core storage").

As you may have already guessed, the programs and data in main storage are in that storage area only temporarily. If they weren't, the computer operator would only need to load a given program once and be done with the punched cards or magnetic medium from which it was input. Any information in primary storage remains there normally only until the data have been processed and the results have been output. Once a run is finished, new data and programs are loaded into the computer and they replace the old program and data in primary storage. (This is the reason programs and data files are kept in permanent form on various media like magnetic tape, magnetic disk, and punched cards for storage in the computer room.)

Programs and data are "stored" temporarily in main storage. You can think of main storage as resembling a post office with a lot of numbered boxes.

**Fig. 4-3** Primary storage with 20 empty boxes



The numbers in the corners of the boxes indicate the location or address of the box, and each box can store only one piece of data. Using location numbers, you can see that a piece of data stored in any box can be "found" simply by calling on its location number.

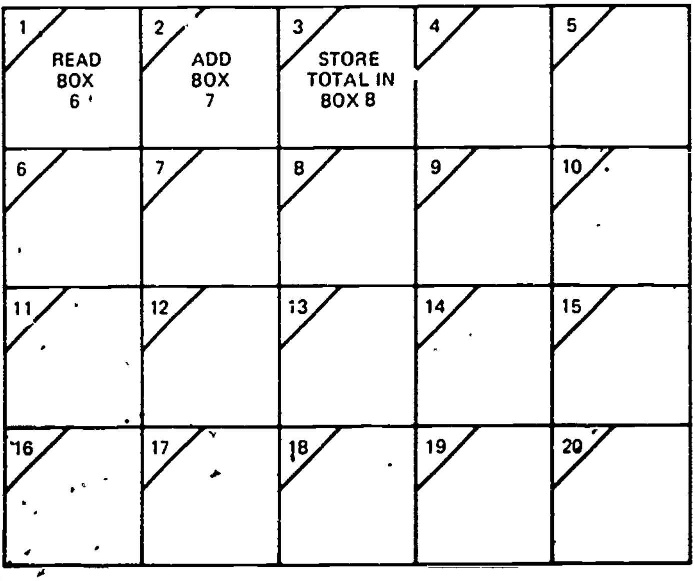
If we continue to use the post office diagram, Fig. 4-4 shows how primary storage might look once it has been loaded with a program for adding two numbers (which will be input later to boxes 6 and 7).

Now, let's assume the two numbers (the data) to be added for our first run are 1143 and 921. They would be input after the program, and primary storage would then look like Fig. 4-5.

Once the data is processed, primary storage would look like the diagram in Fig. 4-6.

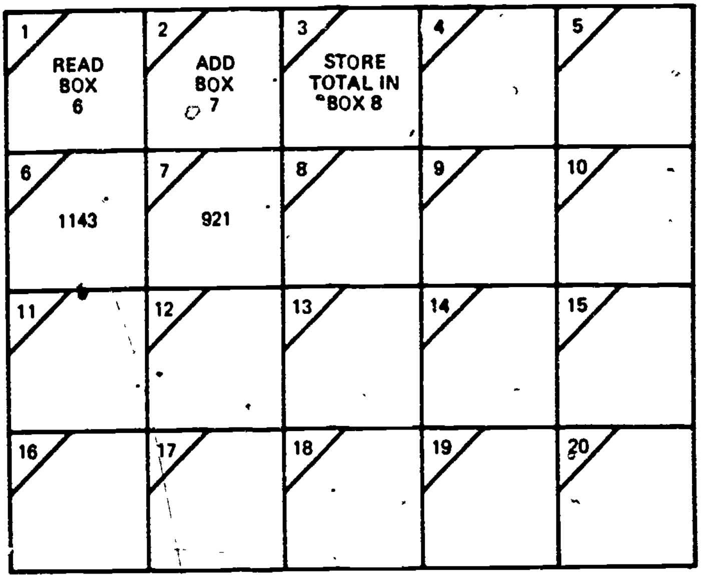
This example is, of course, a terribly simple one. It is just to give you a clear idea of how data are stored in main storage and how they can be found there for use when the processing must be done. Most programs are much longer. Data for a run can be extremely long as, for example, in the case of complete records from kindergarten on for all students in a high school.

Once in primary storage, programs and data can be "accessed" (or found) at any time for use by the computer—that is, until new data have been loaded to replace them in the same primary storage locations for another, separate run. This indicates the temporary nature of primary storage. Once a run has been completed, the program and data



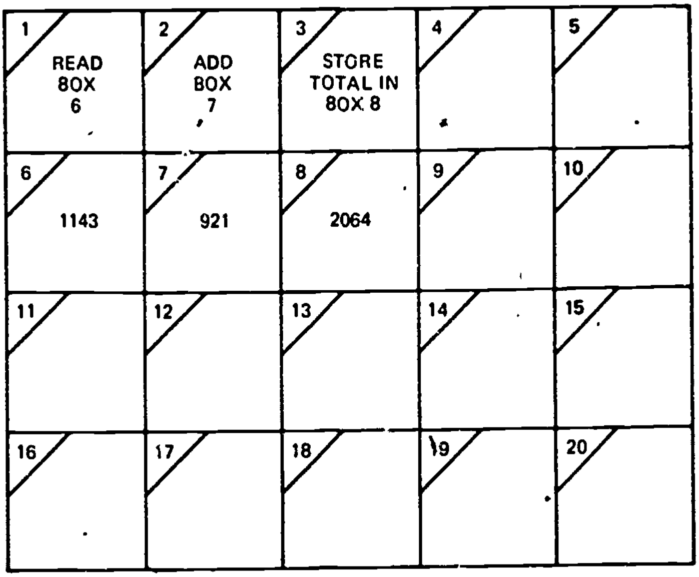
| | | | | |
|---|---|---|---|---|
| 1<br>READ BOX 6 | 2<br>ADD BOX 7 | 3<br>STORE TOTAL IN BOX 8 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 |

**Fig. 4–4** Primary storage with a program in it



| | | | | |
|---|---|---|---|---|
| 1<br>READ<br>BOX<br>6 | 2<br>ADD<br>BOX<br>7 | 3<br>STORE<br>TOTAL IN<br>BOX 8 | 4 | 5 |
| 6<br>1143 | 7<br>921 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 |

**Fig. 4–5** Storage with programs and data



| | | | | |
|---|---|---|---|---|
| 1<br>READ<br>BOX<br>6 | 2<br>ADD<br>BOX<br>7 | 3<br>STORE<br>TOTAL IN<br>BOX 8 | 4 | 5 |
| 6<br>1143 | 7<br>921 | 8<br>2064 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 |

**Fig. 4–6** Primary storage after processing of data

for the next run are loaded into storage and replace (or erase) the data held in those locations before.

Usually the "size" of a computer is determined by the number of these "boxes" or storage locations in its primary storage. People usually refer to a computer with 5 to 20,000 such storage locations as a minicomputer, especially when comparing this capacity with a computer that has, say, half a million (500,000) primary storage locations.

## Arithmetic/logic unit

It is in the arithmetic/logic unit that the real "processing" of data takes place. This unit performs calculations and makes decisions, according to a program stored in memory.

The logical operations that are performed here include comparing information and making a decision based on the results of that comparison. A typical comparison might be

"Is the answer less than zero? (Compare the answer with zero.)
If so, stop the program. If not, go on to the next step."

The arithmetic operations of adding, subtracting, multiplying, and dividing are also performed by this unit. Any complex calculations can always be reduced to a combination of these four operations, so that even the most difficult (for us) calculation is performed effortlessly and automatically by the arithmetic/logic unit of the computer. In the example we showed above in the section on main storage, it is the arithmetic/logic unit which registers the number 1143, adds to it the number 921, and comes up with the total 2064, which is then stored in main storage. Of course, the computer does all arithmetic/logic operations with binary digits. If you would like to know how, you can find out by studying the Appendix.

If primary storage just holds data at prescribed locations and the arithmetic/logic unit performs operations, then how does anything know when or how to go where for what processing? What keeps everything from happening at once? The control unit!

## Control unit

The control unit directs the flow of information through the computer system. It controls the movement of information between input/output devices and primary storage, and directs the sequence in which operations will be carried out. Basically, it simply allows one step of the program to be acted on at a time, one after the other. Using our example from the previous section on primary storage again, it is the control unit which would first indicate that box 1 should be acted on—that is, the number 1143 should be registered by the arithmetic unit. After that, it allows box 2 to be acted on, and so forth, until the program run is completed.

You might think of this unit as acting like the conductor. of a symphony orchestra who follows a musical score (the stored program) and cues the various instruments (input, output, and storage devices) when it's their turn to play. The conductor keeps the orchestra together so that it plays harmoniously in the sequence prescribed by the musical score. The output then is predictable and pleasing, as is the output from an efficiently controlled computer program.

Before going on, check your knowledge of what you have read.

## Check your understanding

1. Assume you have this three-step multiplication program:
    - Read the number in storage location 8
    - Multiply it by the number in location 9
    - Store the product in location 10

    If you were using this program to multiply 8 × 1110, where would the answer 8880 be stored in primary storage?
2. Are all programs and data stored permanently in the CPU's primary storage? Why or why not?
3. The process of multiplying 8 × 1110 is performed by what part of the CPU?
4. Which part of the CPU is responsible for coordinating the sequence of operations inside the CPU?

### SECONDARY STORAGE DEVICES

You will remember that the information in primary storage is temporary. That is, it remains in primary storage only long enough to be processed and output. Then new data are input to primary storage, and the old are erased. (This is very much like a home tape recorder. When you record a new song, whatever was on the tape before is automatically erased.)

Frequently, however, we have a need to store a large volume of information, and to store it permanently. For example, the complete pupil records from kindergarten on for every student in a high school would be too much information to be stored permanently in primary storage. Even if a computer had a large enough primary storage unit, it would still need to be kept available for the temporary storage of programs and other data currently being processed.

Secondary storage devices solve this problem. They can be connected to the CPU to provide information from large permanent storage files. Since the control unit has to go outside the CPU to get data from the secondary storage devices, the information is not as quickly accessible as information in primary storage. The loss in speed, however, is offset by the larger capacity and the lower cost of secondary storage as compared to primary storage.

Many of the input/output devices you have studied about can be used as secondary storage devices as well. Magnetic tape, disk, and drum units are the most common. Even punched cards and paper tape can be considered as "secondary storage" in that they permanently store information outside the CPU.

There are other special secondary storage devices which can hold much more information than tapes, disks, or drums. One new type of computer storage device may replace all storage devices currently being used to store more data, and retrieve them faster, more reliably, and at less cost. This new device is "holographic" memory and is based on the laser beam. It stores data in "holograms" formed by laser beams on a thermoplastic medium in the form of miniscule light or dark areas—the zeros and ones of the binary code of computers. This new type of memory can store as much information as the largest magnetic disk systems available today, but store and retrieve it 1000 times faster.

### Primary, secondary, and on-line storage

You might think of the way data are stored in a computer system as being similar to the way you store (or remember) data yourself. Some data you store in your brain, available for immediate processing. An example of data of this type would be your own phone number—and perhaps the phone numbers of several good friends. This kind of immediate-access storage in a computer is the primary storage unit in the CPU.

Other telephone numbers which are important to you, but you use less frequently, you might store in a quickly-accessible place. But you wouldn't store them in your memory. Instead, you might write them down in your pocket address book or on a piece of paper to keep in your wallet, handy when you need it. This kind of storage would be similar to *on-line* secondary storage in a computer—that is, magnetic tapes or disks or some other medium that is on-line (directly connected) to the CPU, so that the CPU can quickly locate information in secondary storage when it is needed.

Finally, there are thousands of telephone numbers you use rarely, if at all. It would be cumbersome or impossible to carry these numbers around in your head or in your pocket. So they are stored in a place you can get to if you need it—in the telephone book on the shelf. This is similar to *off-line* secondary storage, such as punched cards stored in a drawer, or magnetic tapes or disks stored on a shelf or rack. The data are accessible, if needed, but are not directly connected to a computer. It is important to note that off-line storage can become on-line storage. If a reel of magnetic tape stored off-line on a rack is placed on a magnetic tape drive connected to the computer, it is now on-line storage.

## Check your understanding

1. Name five secondary storage devices.
2. If your brain is your primary storage unit, written notes are your secondary storage unit, and library books are your off-line storage, which type of storage unit do you use to store your home address in?
3. In which type of storage unit do you store the number of people in your home town?
4. Which type of unit stores the dates of the Civil War?
5. In your school district's computer center, which kind of storage do you suppose stores your grades since you were in first grade?

### COMPUTER CONSOLE

One of the major values of a computer is that, once started working on a job, it can basically proceed to the end without human intervention. A stored program provides the machine with the instructions it needs to obtain information, process it, and output new information.

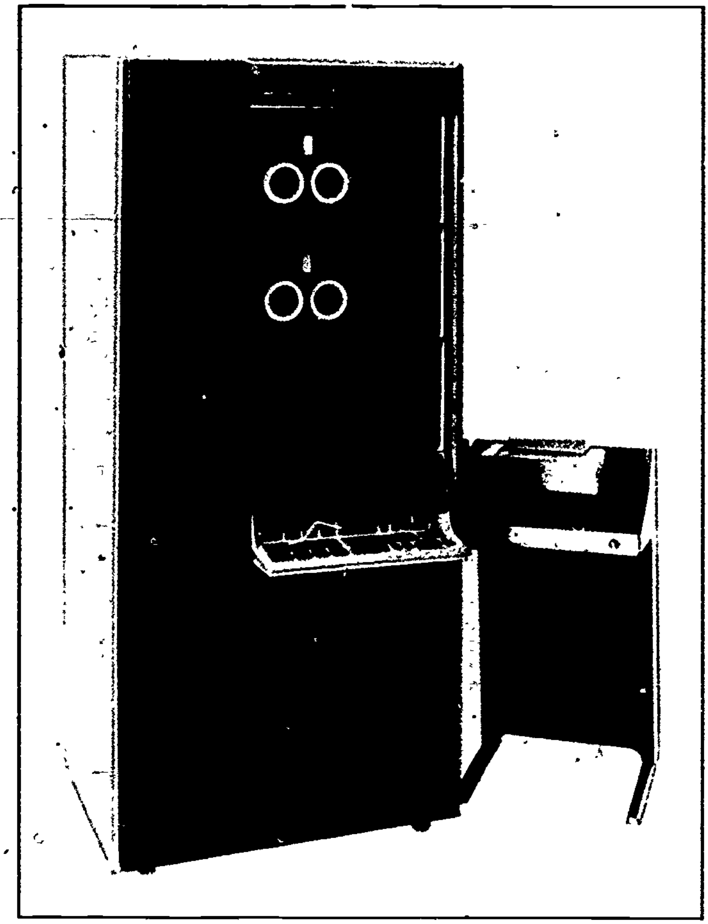


Fig. 4–7 A Digital PDP-12 computer console with control panel and console typewriter

In practice, of course, the operator must communicate with the computer in order to get the computer ready for running a program. Occasionally, the operator must intervene during the data processing.

As you have seen, there is a wide variety of input-output devices available for exchanging information between a computer and its human directors. All of these devices allow an indirect kind of communication. The only direct communication with the computer is via the *computer console*, found on the CPU.

Each computer console model has its own special features, but the typical console contains two devices for manual communication with the computer. These two devices are the console typewriter and the control panel.

## Console typewriter

The console typewriter looks like an ordinary typewriter, with a standard keyboard and paper output. This typewriter can be used both for output from the computer and input to it.

At the direction of the program being run, the computer can type out messages tq the operator on the console typewriter:

"MOUNT TAPE REEL #4"
or "PLEASE TYPE TODAY'S DATE"
or "WHAT SIZE LABELS?"
or "HOW MANY COPIES ARE NEEDED?"

The operator can answer these queries from the computer by typing on the keyboard. When a key is pressed, the corresponding number, letter, or symbol is typed on the typewriter paper and is simultaneously transmitted into the computer in electronic code.

The console typewriter could actually be used by the operator to input any kind of material into the computer—data, programs, or the operating instructions called *job control statements*. But, can you see why it usually isn't used for these things? First, when data or programs have once been encoded into cards, tapes, or other media, they can be kept indefinitely. When they are needed for use, the medium can be mounted and loaded into the computer in a matter of minutes—even seconds! If you input a program or data file from a console typewriter, it could take a great deal of time. Not only that, but you would have to retype it on the keyboard every time you wanted to use it. Second, the instructions given to the computer in the form of job control statements are usually basic directions needed by the computer to properly run the program involved. Since one set of job control statements is always needed for the running of each program, it is more convenient to have the set of statements on cards or paper tape for instantaneous input than to have them typed on the console typewriter each time the program is to be run.

Generally speaking, the console typewriter is the computer operator's main means of communicating with the computer to *control* its operation. The main input is therefore instructions for control purposes.

Commonly, the operator uses the console typewriter initially to instruct the computer to get ready for a specific run and then to activate the input devices to "read in" appropriate programs and data. Further, during the processing of data when the program calls for responses from the operator such as in the examples above, the computer operator types in the answer on the console typewriter. Sometimes instructions must be input by the operator in mid-run. These are always clearly noted on the instruction sheet given to the operator by the programmer, and the information is input from the console typewriter. Finally, the operator uses the console typewriter to instruct the computer to end a job either when a problem has arisen or when the job is done. The console typewriter can also be used to directly examine and to alter the contents of any storage location. The procedure involves simply typing in a request that/the data in a specific location be typed out by the computer and then typing in any alterations to be stored.

Each computer has an operator's manual or directions booklet which outlines the procedures the operator must use in controlling routine runs from the console typewriter.

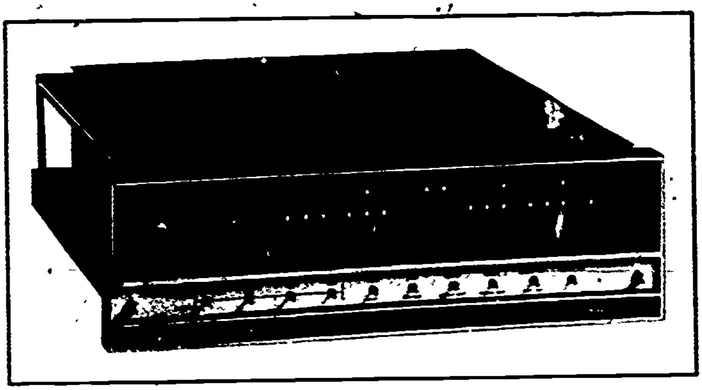
### Control panel

The control panel is like a window into the computer. It has sets of small incandescent lights which display the contents of certain internal registers. It also has a means for manually displayirg, examining, and altering the contents of storage.

While the computer is running a program, the lights change too rapidly for the contents to be read. Whenever the computer stops because of a problem, though, the light display showirg the register contents is often a valuable aid in determining what has happened.

The control panel is occasionally used for detecting program mistakes—a process usually called "debugging." An operator can set switches on the control panel to stop operations in order to read the content of a storage location, observe a program instruction, or examine the registers of the arithmetic unit after each instruction. This kind of debugging by step-by-step operation and observation of the program is, however, usually ruled cut as inefficient in a large computer system.

The computer operator also uses the control panel to monitor the general operation of the system. Indicator lights on the control panel show the operation or condition of important parts of the CPU and in such equipment as the card reader, magnetic tape units, and the printer. For example, there may be a temperature warning light, overflow lights, error indicators, and lights to signal some off-normal condition, such as the printer being out of paper.



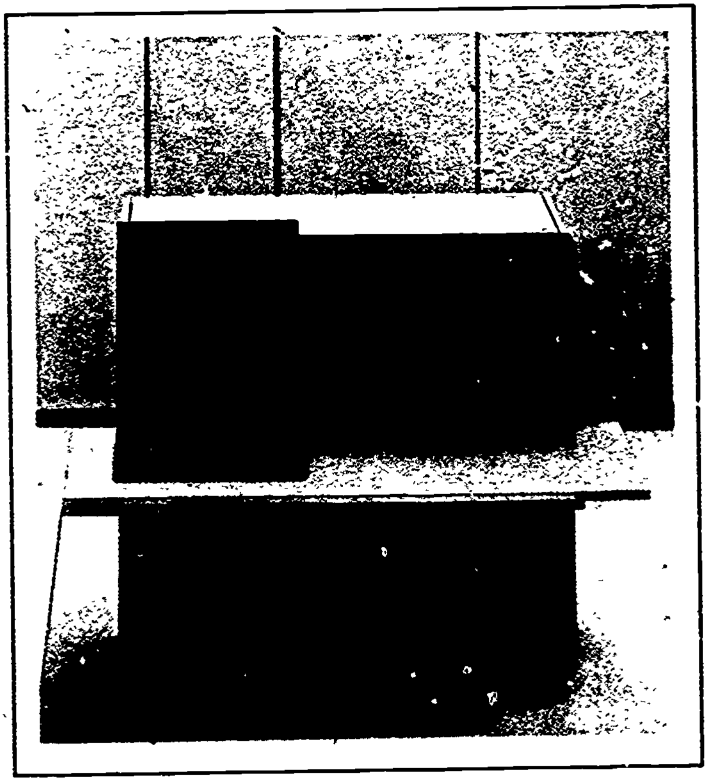


**Fig. 4-8** Control panels with characteristic rows of indicator lights and control buttons. Control panel on a Data General Nova 1210 (top), control panel on an IBM 370 (bottom)

Probably the one switch always found on a control panel is the START key. When the START key is depressed, it causes the computer to start running a program or to continue after an operator intervention.

Between the console typewriter and the control panel, a computer operator can stay in very close communication and control of the computer he or she is responsible for.

# Check your understanding

1. Primary storage in the CPU differs from secondary storage by

   a. being less permanent.
   b. having less space available.
   c. being more immediately accessible to the computer.
   d. all of the above.

2. What happens to old information in a primary storage location when new information is placed in it?
3. The part of the CPU which acts as the monitor of the system's operation is

   a. the control panel.
   b. the logic unit.
   c. the "mainframe."
   d. the console typewriter.

4. The arithmetic/logic unit

   a. keeps track of all data in storage.
   b. can perform basic arithmetic operations.
   c. controls the movement of all data to and from main storage.
   d. All of the above.

5. The console typewriter is

   a. often substituted for a control panel.
   b. used as a general input device.
   c. part of auxiliary storage.
   d. the operator's main device for communication with the computer.

## INTRODUCTION TO SOFTWARE

Computer hardware is the most visible part of a computer system, but it is not the most important. Even more important than the hardware is the software.

Software is the word used to refer to all of the programs that make the computer function. Without software, a computer is like a camera without film—all the button-pushing and switch-flipping on earth won't produce anything.

Early computers used very simple programs. A computer programmer would write a set of instructions in a numeric code telling the computer how to solve a problem or process some data. Then the computer operator would take over. First, he or she would load the program from punched cards or some other medium into the computer. Then, the operator would put the data in an input device—usually a card reader—and start the program run. During the run the operator might have to interrupt the operation of the computer several times—to provide more data, to mount new magnetic tapes for output, to prepare the printer for output, to check the output for accuracy, and even, perhaps, to make manual charges in the program itself by operating the control panel switches.

When one job (or program run) was completed, the operator would stop the computer, retrieve the data cards used, gather up all the output, and prepare for the next job. Meanwhile, the computer waited. Then the process would begin again. Load a program, provide data for input, control and keep track of the input and output, and manually monitor the operation of the computer during the program run.

As you can see, early computers were inactive a lot of time both during and between program runs while waiting for the computer operator to performs tasks.

Today's computers are far closer to being true "labor-saving" devices. Sophisticated developments in hardware have been matched by an increase in sophistication of the computer's software. Nowadays many of the taks of the computer operator have been taken over by software, so that jobs can be run more quickly and efficiently and automatically, with less waiting time for the computer, and less intervention by the computer operator.

Just as developments in software have increased efficiency of computer use and lightened the computer operator's load, so have software developments made the job of the programmer easier.

It used to be necessary that programs be written in the specific numeric code that the given computer used. Now, though the computer still works with numeric code, programmers write instructions in ordinary English words. New software can now translate these instructions into the required numeric code, saving much time and trouble for the programmer.

When the decision is made to acquire a computer system, it is as important to select a system with good software as it is to select good hardware. A computer without proper software will add considerably to the cost of operating the installation. Software with low efficiency may increase computer running time by 25 percent or more.

The manufacturer usually supplies a complete "software package." It will include special programs to make the computer operate most efficiently, programs to do standard, often-used routines, and translators which will convert new programs into the machine's numeric code. This software package is ordinarily called the *operating system.* The programs in it are usually stored on an auxiliary storage device such as a disk or drum.

Computer software includes other categories of programs in addi-

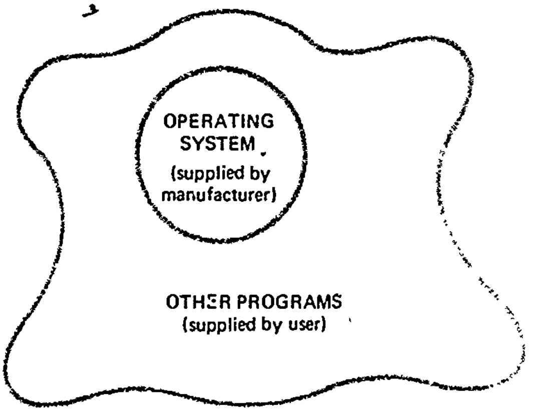


**Fig. 4–9** The computer software

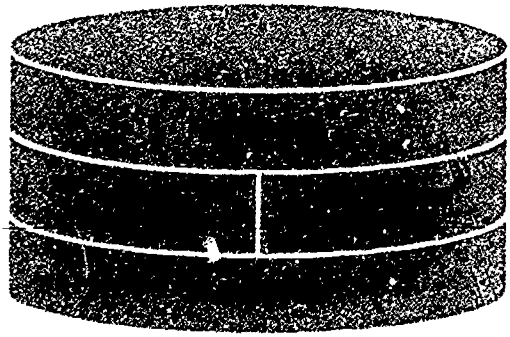


**Fig. 4–10** Operating system stored on an auxiliary storage device

tion to the operating system. These programs outside the operating system are usually designed by users to perform specific jobs to meet their particular needs; therefore, no general summary could hope to discuss them all. However, Fig. 4-9 is a more detailed examination of the operating system, which in one variation or another is found in every software system.

## The operating system

An operating system is an organized collection of programs which increases the productivity of a computer by aiding in the preparation, translation, loading, and execution of programs. Intervention by the computer operator is reduced when an operating system is available. This usually means that an operator can use the time during one run to set up the equipment for the next run. Often, several jobs can be run in an uninterrupted sequence, under the control of the operating system.

The programs most likely to be found in an operating system will be discussed here. However, the programs available may vary from one operating system to another. Certain programs may be called by other names than those used here. This chapter should be a starting point. When you have an opportunity to operate a real computer system, you should carefully read the manuals that come with the system. These manuals usually give complete descriptions of the software included.

The types of programs most often available in an operating system include the following:

- A supervisor (often called *monitor* or *executive*)
- Input/output control
- Utility programs
- Translators

Now let's look at these programs one at a time to see just how they increase efficiency and ease of operation for not only the computer, but the programmer and operator as well.

***The Supervisor.*** The supervisor program or routine is the most important and complex of all programs in the operating system, because it sets up, monitors, and controls the operation of other programs. While most software is placed in auxiliary storage (disk or drum) until needed, the supervisor routine is loaded into the computer each morning. It usually remains in main storage throughout the day's operation. It organizes and supervises the running of all other programs in the operating system.

A basic supervisor routine controls the execution of one program by clearing the storage area (placing zeroes or blanks in work areas), then loading the program into storage.

Once the program is running, other programs in the operating system take care of the flow of operation. When errors are encountered during processing—for example, when a program exceeds the storage capacity—control is once again transferred to the supervisor routine which takes corrective action or halts execution.

An important part of the supervisor routine is the "job control" program. This program handles a series of programs sequentially with a minimum of operator intervention. To do this, the computer requires initial instructions. The programmer usually has these instructions punched on cards and gives them to the computer operator for the run. These cards are called the job control cards. The instructions on the job control cards may give the sequence of steps to be performed and may indicate what programs in the operating system are to be used.

The supervisor routine starts and stops the programs and performs many functions normally otherwise done by the operator, but it does them faster and more accurately. Where operator intervention is required, the supervisor will print out operator instructions on the console typewriter.

In some more complex, modern computer systems, the central processor is a *multiprocessor*. That is, it has more than one arithmetic-logical unit. This means that several programs can be run at the same time. Of course, the supervisor routine and its job control programs must be even more sophisticated to schedule and monitor the execution of more than one program at once.

In other systems, a technique called *multiprogramming* is used, where there is only one arithmetic-logic unit, but several programs are in the computer at one time. The supervisor routine divides the computing time among the different programs, transferring control from one to another to process them all at the same time. This technique is called time-sharing. Again, a sophisticated supervisor routine is required to schedule and control the concurrent execution of the programs.

As an example of multiprogramming, a program to read and print cards may be run concurrently with a program to sort records on magnetic tapes. The supervisor routine initiates the instruction to "read a card" for the first program, then it may transfer control to the sort program. When a signal indicates the card has been read, the second program would be interrupted and control returned to the read-print program for moving the data into position for printing. Since the read-print program can proceed no faster than the reader and printer can operate, the CPU would be idle much of the time if the supervisor routine did not use this idle time to run another program.

*Input/Output Control Programs.* The control of input and output (I/O) operations requires a separate set of programs because these are mechanical functions performed by mechanical devices. If the speedy electronic central processor had to stop and wait during each I/O operation, its efficiency would be greatly reduced.

If data were read directly into the CPU, or directly out, the fast CPU would be tied up as the slow mechanical reader or printer operated. However, most computers now use electronic buffers to cut down this delay. When data are read from an input device, they go into an input buffer. They are held there until the CPU is ready for them. Then they are transferred from the buffer to the CPU to the output buffer and output when the output device is ready.

The scheduling and overlapping of these buffered I/O operations is complicated and must be controlled by the input/output control programs. These control programs also handle such operations as detecting I/O errors, switching to an alternate tape drive at the end of a reel, and searching for the proper items to be input from a reel of tape, a disk, or a drum.

*Utility programs.* Utility programs support the production work of a computer center by servicing jobs which recur frequently. While the specific utility programs available will differ from one system to another, a fairly complete set would include the following:

- A program loader
- Programs to transfer data from one medium to another (card to tape, tape to card, card to disk, disk to card, tape to printer, disk to printer, etc.)

- "Diagnostics" or test routines to help identify and correct machine failures or program errors
- Special programs to protect permanent files, sort and merge data, duplicate tapes or disks onto new tapes or disks, and perform "memory dumps" (output the entire contents of main storage on the printer)

The program loader is used to enter new programs into storage. Even the programs in the operating system must be initially introduced into storage by the loader. A program is stored in memory in the same way as data. It is read in, probably from punched cards or tape. The loader supervises the reading in, allocates storage locations for each program instruction, and finally transfers control to the first instruction of the program just loaded, so the program can be executed.

Transferring of data from one medium to another is often necessary. It must be controlled by special programs such as card-to-tape or tape-to-printer routines. Likewise, there are special programs for duplicating tapes or disks onto new tapes or disks.

Diagnostics or test routines may function automatically to check for machine or program error during operation of programs. They may also be called into action when there appears to be a malfunction in a program. If a programmer wishes to use a test routine designed to help locate program error, she or he would probably call for a "trace" or a "dump." A trace routine will print the results as each step of the program is executed, to allow the programmer to closely observe the operation of the program. A dump routine, often called a memory dump

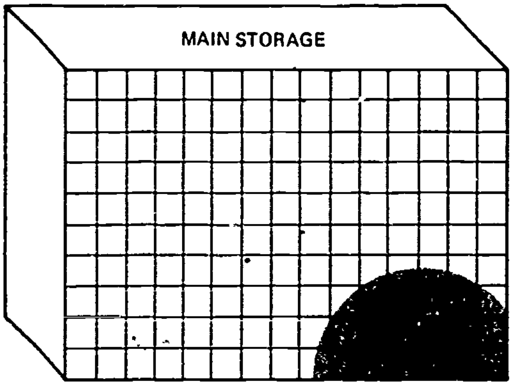


Fig. 4–11 Program loader in main storage

or memory print, will print the contents of each storage location at any specified checkpoint in a program. The programmer can also specify the boundaries of the storage area she or he wishes to examine.

Special programs take care of frequently performed operations, such as copying all files from tape or disk at intervals as a protection against accidental erasure or loss of important data. Another frequently used utility program found in most systems is a sort-merge routine. It should be able to sort either numeric or alphanumeric records into ascending or descending order, merge one set of records with another set of records, add and delete data, and provide for input and output on various media.

***Translators.*** In the early days of computers, programmers wrote instructions entirely in a numeric code directly understandable by the computer. This meant that the programmer had to translate each instruction to a corresponding numeric code.

For example, if a programmer wished to write the program instruction

"SUBTRACT TOTAL DEDUCTIONS FROM GROSS PAY"

he or she would first translate it to a code that might look like this:

07 326 145

where:

07 = "SUBTRACT"
326 = The address of the location where "TOTAL DEDUCTIONS" is stored
145 = The address of the location where "GROSS PAY" is stored

As you can see, the job of the programmer could be quite time-consuming, tedious, and error-prone.

But computers were invented to save work for humans, not to create more! It wasn't long before some programmer with writer's cramp realized that a sophisticated translator program could automatically translate an instruction like

"SUBTRACT TOTAL DEDUCTIONS FROM GROSS PAY"

to the numeric code

"07 326 145"

much faster and more accurately than the programmer could translate it himself.

Since computers are designed especially to manipulate symbols (like letters and numbers), translating words into numeric codes is much easier for the computer than it is for humans. Although computers still operate exclusively in numeric codes, seldom does a programmer have to actually use them now. Over the years, several different kinds of translators have been developed to ease the task of the programmer.

The most simple kind of translator is called an assembler. It requires more of the translating effort to be done by the programmer and less by the computer. Most operating systems include ai least one assembler.

The type of translator used most often by programmers is the compiler. Compilers allow the programmer to write instructions in a language much like conversational English. The translating to numeric code is almost completely carried out by the computer.

For example, the programmer can write a single instruction like:

```
"GROSS PAY — (SOCIAL SECURITY + OTHER DEDUCTIONS)
                    = NET PAY"
```

and the compiler will "compile" or translate this single instruction to a series of several numeric codes, such as the following:

```
06 325 324
15 326 000
07 326 145
15 327 000
```

One common compiler is used primarily to translate programs used in business. It is called COBOL: COmmon Business Oriented Language. Another widely used compiler is called FORTRAN, for FORmula TRANslator. FORTRAN was designed to translate scientific programs to numeric code.

BASIC (Beginner's All-Purpose Symbolic Instruction Code) is a compiler used primarily with remote terminals, where the user carries on an interactive "conversation" with the computer. Any or all of these compilers, and perhaps other compilers as well, may be found in the operating system supplied by the computer manufacturer.

Some operating systems include special-purpose translators that cannot be called either assemblers or compilers. They may be called report program generators or interpreters, but they still serve as translators of our language to the machine's numeric code.

Now see how well you understand the basics of the computer's software by answering the following questions.

## Check your understanding

1. The software of a computer is

   a. the electrical circuitry of a system.
   b. the documentation of a system.
   c. the computer programs.
   d. the data cards, tapes, etc.

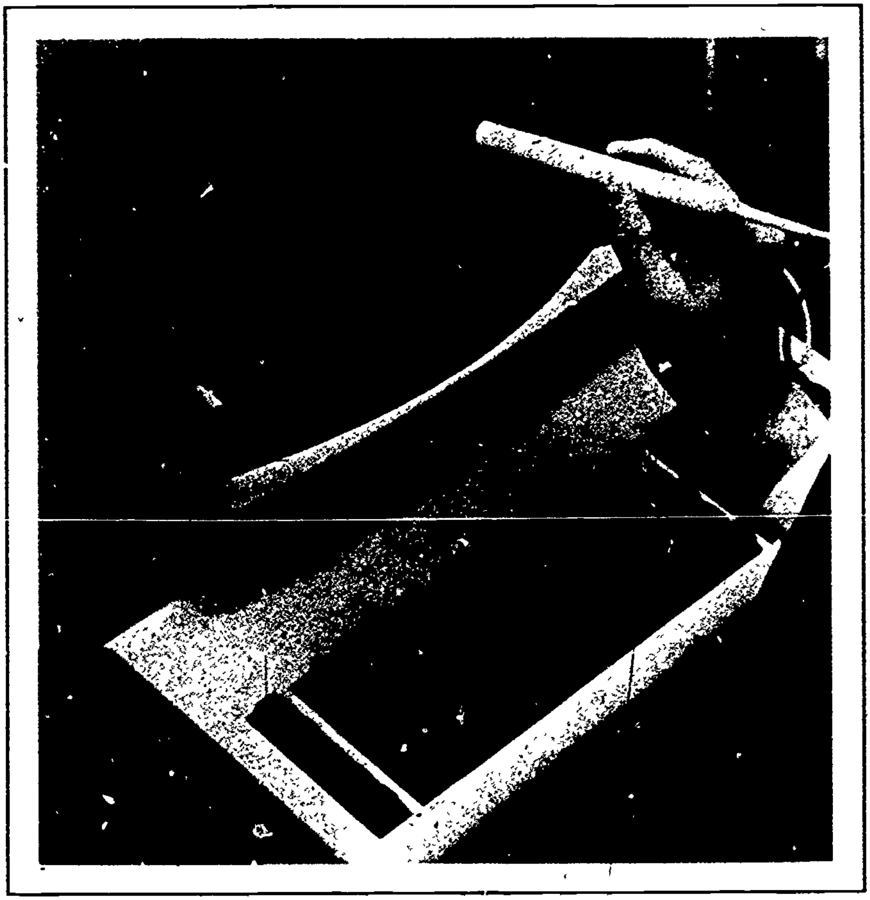
2. A translator is

   a. a programmer specializing in program languages.
   b. a program used to control input and output.
   c. a program which converts programs from words to numeric codes.
   d. none of the above.

3. Assemblers and compilers are both

   a. input devices.
   b. programmers.
   c. utility programs.
   d. translators.

1234567



# Computer programming: techniques and tools

## INTRODUCTION

Applications computer programmers have one of the most detailed jobs in the computer center. They must flowchart programs and code them in a specific language. They must test and debug their programs. Then, finally, applications computer programmers must document their programs for use by the computer center.

Programmers have several techniques by which they can plan, code, and then test a program. They also have basic tools in the programming process which make efficient use of computer time and storage. This chapter will give you some understanding of the tools and techniques used by computer programmers to implement a computer program.

## FLOWCHARTING

Obviously, the best way to solve a problem, particularly a complex one, is to break it down into steps and to arrange the steps in an ordered sequence

Some computer problems may be simple enough that all the steps are quite clear. In more complex problems, the sequence may be harder to see. To deal with more complex problems, a technique has been developed called flowcharting. It involves the use of symbols which chart each step in solving a problem. Specifically, a flowchart can show step-by-step what the computer must do to solve a computer problem.

### The structure of a flowchart

It is important to know that a flowchart almost always has a beginning, a middle, and an end. The flowchart helps programmer to see the solution to a computer problem graphically by showing exactly what information must go into the computer in the problem, what the computer must do to process the information, and what the end result will be

In solving a computer problem, a programmer deals primarily with three elements. They are the beginning (or information needed to solve the problem, called input), the middle (or process), and the end result (called output). If you were to flowchart the three basic elements in a computer problem, they might be drawn as shown in Fig. 5-1.

As you can see, flow lines are drawn from one flowchart symbol to another. They show the sequence and direction of events. Although these symbols are drawn vertically, flowcharts may indicate the same steps if they are drawn horizontally, or across the page.

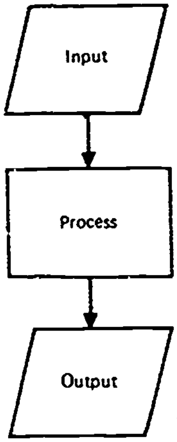


Fig. 5–1 Flowchart of the three basic elements in a computer problem

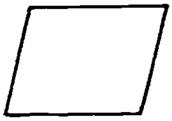


Fig. 5–2 Input/output symbol

## Flowcharting symbols

Different flowcharting symbols are used to show particular actions to be taken by the computer. You can see below how each symbol describes each computer task. First, if the computer is to input or output data, the action is shown by an input/output symbol.

If a decision about data must be made, the flowchart will contain a decision box with lines for at least two results of the decision, called "alternatives."

The "No" branch coming out of the decision symbol in Fig. 5-3 leads into a small circle which is called a "connector" symbol. Sometimes flowcharts get so complex that flow lines cross each other and make the logic hard to follow; or the flowchart may be so long as to extend over several pages. Using the connector, a programmer can eliminate confusing lines, and also indicate where the flow reconnects to the chart on other pages.

Any actions to be taken upon data brought into the program are shown by a process box. These may include actions such as moving data

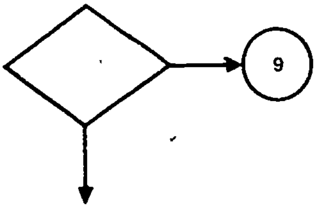


**Fig. 5–3** Decision symbol and connector symbol

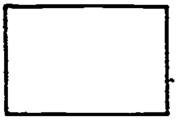


**Fig. 5–4** Process symbol

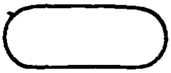


**Fig. 5–5** Terminal symbol (start or stop)

around in storage or performing an arithmetic operation. The process box looks like that shown in Fig. 5-4.

An instruction to start a program or to stop a program at the end of processing is illustrated in a flowchart by a terminal symbol, as in Fig. 5-5.

## Refining the flowchart

When making more detailed flowcharts of computer solutions to actual problems, programmers often use input/output symbols which represent a specific kind of input or output device. As you can see from Fig. 5-6, there is an input or output symbol for each major device on the computer.

By using these symbols, a programmer can show graphically exactly what form the input to the computer will take. The programmer can also show in what form the results will be produced.

Actually, when computer programmers flowchart a problem, they describe their input or output in even more detail than you have seen so far. As you know, input can be *read* into the computer and output can be *written.* A programmer, in refining the steps of the problem into a more detailed flowchart, may actually name the input and output, using

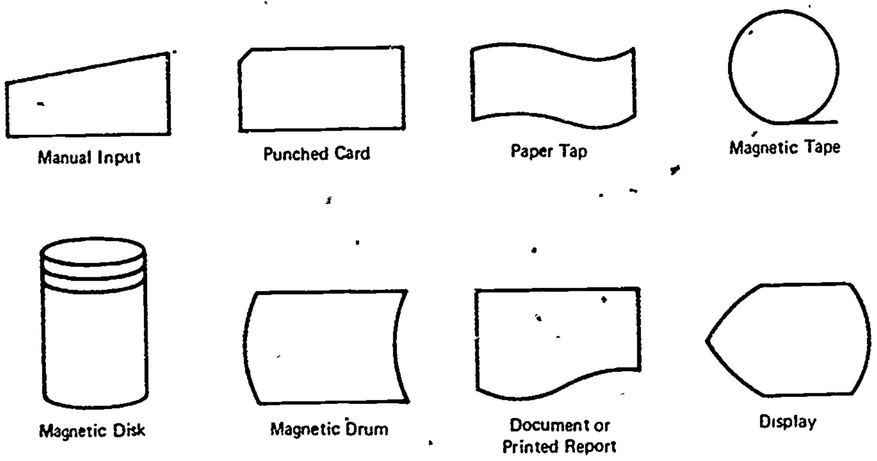


**Fig. 5–6** Specific input/output symbols

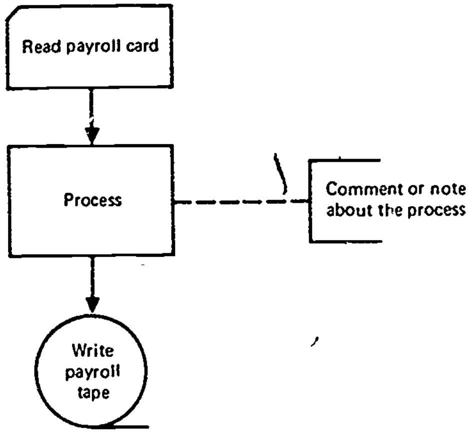


**Fig. 5–7** Flowchart symbols

specific flowcharting symbols to describe them. The illustration in Fig. 5-7 shows you how a flowchart can describe data in detail.

Figure 5-7 shows an additional way in which a programmer can describe what is happening in a flowchart. The three-sided box joined to the flowchart with a broken line is called a comment symbol. Descriptive comments or explanatory notes can be added at any point in the flowchart by using this symbol. The broken line may be connected to any other symbol wherever it is appropriate.

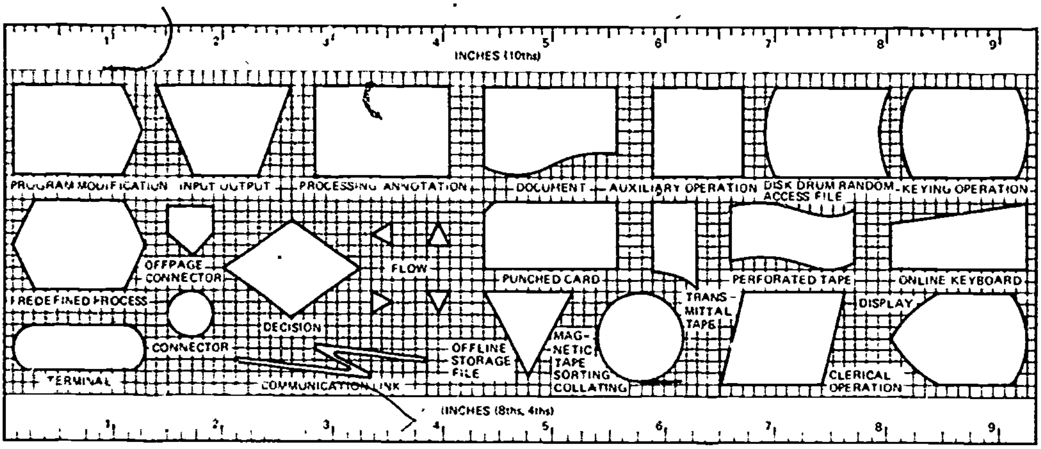


Fig. 5–8 A flowcharting template

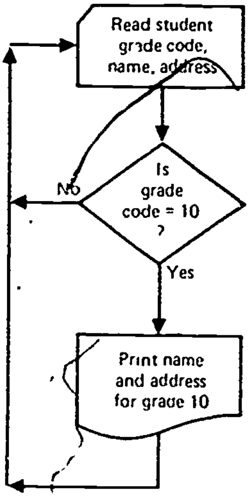

Read student grade code, name, address
Is grade code = 10 ?
No
Yes
Print name and address for grade 10

Fig. 5–9 A complete flowchart

You are probably wondering how the symbols for flowcharts are made. The answer is that programmers use a special tool called a template which is a plastic plate with the symbols' shapes cut out of it. You can see an illustration of a programmer's template in Fig. 5-8.

You probably noticed there are symbols on the template you have not yet seen. The symbols which have been described so far, however, are really the ones you need to know to solve problems in this section. Later, if you pursue your study of programming, you will find more and more uses for all the flowcharting symbols.

Now look at the example in Fig. 5-9 of a flowchart which uses several of the symbols you have been studying.

As you can see, this flowchart illustrates the simple process of reading data from punched cards and printing student names and addresses. Did you notice that only certain students' names and addresses are to be printed? (Only the names of students with grade code "10" are to be printed.) Any data with a grade code other than "10" will be ignored. Notice that after each card is read (whether its grade code is equal to "10" or not), the flow lines in the flowchart return to the "read card" symbol. This indicates that the processing will continue until all the cards have been read. You can see how easy it can be to diagram the solution to a problem using the flowcharting technique.

Before you go on to learn to program, check your understanding of flowcharting by answering the following questions.

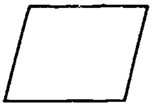
# Check your understanding

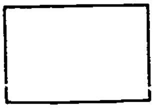
1. Flowcharting is a technique programmers use to

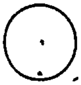
   a. solve only complex computer problems.
   b. arrange steps of a problem in an ordered sequence.
   c. choose one alternative over another.
   d. solve only easy computer problems.

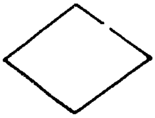
2. Identify the following symbols by using the list of names at the left.

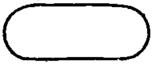
1. Terminal
2. Process box
3. Decision box
4. Input/output
5. Connector

________ 

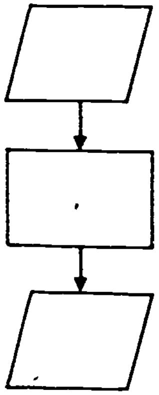
________ 

________ 

________ 

________ 

3. Copy the flowchart below. Fill in on your copy the three elements present in any flowchart of a computer problem.

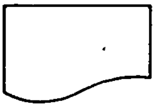


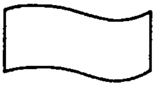
4. In a detailed flowchart, input and output are usually shown

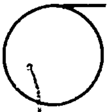
a. by using an input/output symbol.
b. by using specific input or output symbols.
c. by showing the beginning, middle, and end of the problem.
d. none of the above.

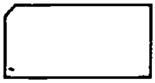
5. See if you can identify each specific input or output symbol shown below. Use the list of names you see at the left.

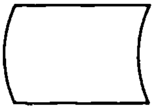
1. Magnetic disk or drum
2. Paper tape
3. Punched card
4. Magnetic tape
5. Printed report











## PROGRAMMING LANGUAGES

You know that a program is a set of instructions which tells a computer what to do. Once a program is stored in the computer's memory, it can act upon data to produce a result. In recent years, programming languages have become so highly developed that there may be one available for almost any kind of computer problem. Programming languages range from the more specific machine-oriented languages to those which can be coded by someone with very little knowledge of any specific computer.

### Machine language

The set of symbols which the computer itself can understand is called machine language. Because computer hardware is designed to accept binary symbols, most machine language is really a binary code (a code using only numbers 1 and 0). The instruction you see illustrated below shows how a typical machine language instruction might look inside the computer.

```
0001 0110     0001 0010     0111 1001
```

However, the computer programmer would simplify the job by writing this code in "shorthand"—a numeric code—as

```
16     12     79
```

This long and involved instruction may be telling the computer simply to store a piece of data in a location in the computer's memory or to read the next piece of data. Computer experts learned quickly that machine language would be impractical for complex computer problems since most programs are made up of many instructions. If every program were written in numeric code, you can see that the programmer's job could become quite tedious.

### Assembly language

In an effort to make the programming task easie , computer experts decided instructions could be written using readable symbols to represent the particular operations of the computer. These symbols, called mnemonic codes or operation codes, were grouped with an "address" to make a complete instruction. The language of operation codes and addresses is called assembly language. Look at the illustration of assembly language instructions below.

| *Operation Codes* | *Address* |
|---|---|
| ADD | 1202 |
| SUB, | 1209 |
| DVH | 2700 |

Can you see that assembly language is more compact for coding many instructions than machine language? Once an assembly language program is coded by the programmer, it can be translated into machine language by the computer.

### Procedure-oriented language

As computers became more complex, programmers saw the need for more usable languages. Larger and larger programs were being written, and many kinds of programming routines (for example, the calculation of the square root of a number) were being written over and over by different programmers. Programmers as a group began to see that they could share their efforts. They also thought a natural shorthand version of their own language would make programming languages more universal.

Eventually, machine language and assembly language were replaced, as programming languages, by higher-level languages called "procedure-oriented languages." Coding became higher-level because, as languages became more like the programmer's own language, the computer had to do more to translate them. For example, rather than programming in numeric code or assembly operation codes, the programmer could use high-level instructions like "MULTIPLY LOAN BY .10 GIVING INTEREST" or "PRINT A + B."

Unlike assembly code, where one instruction is translated into one machine code, these procedure-oriented languages tend to group several separate operations into one instruction or "procedure." For example, the single instruction "A = B + C − D" may be translated into four or five separate machine language instructions.

There are several advantages to using a high-level programming language. Being "shorthand" versions of our own language, they are easier to learn. More attention can be put to the logic of solving a problem rather than to learning a complex code. As you saw in the previous example, procedure-oriented languages are easier to use than machine code or assembly language. Some languages actually use complete sentences in describing an operation.

A problem written in a high-level language is also easier to correct or "debug." There are fewer instructions in high-level language coding than in machine or assembly language, making a program physically shorter. Because high-level languages are easier to read, errors in logic occur less frequently and can be located more readily.

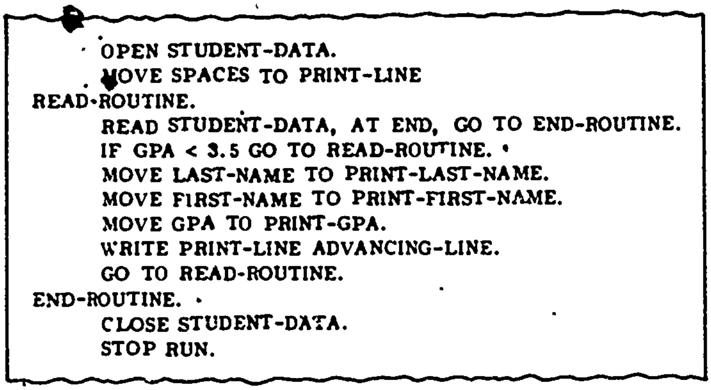
Another advantage to programming in high-level languages is that it is easier to keep programs up to date if they are written in high-level language. In assembly language, when one instruction is changed, other instructions in the program may be affected. In such cases, the programmer must be careful to alter everything which might be affected by one instruction change. With programs in high-level languages, the programmer needs only to make the initial changes in instructions. Any instructions affected by that initial change will automatically be corrected by the compiler program (a part of the computer software).

There are many different procedure-oriented languages designed to solve different kinds of problems. In the next section, you will get a preview of three of the most commonly used procedure-oriented languages—COBOL, FORTRAN, and BASIC.

## Commonly used high-level languages

Three of the high-level languages most commonly used by programmers are COBOL, FORTRAN, and BASIC, each of which has its own special use. Let's look at them one at a time so that you can get an idea of how each is used and how they differ from each other. Later in this manual you will delve more deeply into one of these languages, BASIC, but for now you need only to gain a general idea of what the languages look like and what they can do.

In 1959, representatives from several computer companies met to discuss the need for a computer language especially oriented toward



```
       OPEN STUDENT-DATA.
       MOVE SPACES TO PRINT-LINE
READ-ROUTINE.
       READ STUDENT-DATA, AT END, GO TO END-ROUTINE.
       IF GPA < 3.5 GO TO READ-ROUTINE.
       MOVE LAST-NAME TO PRINT-LAST-NAME.
       MOVE FIRST-NAME TO PRINT-FIRST-NAME.
       MOVE GPA TO PRINT-GPA.
       WRITE PRINT-LINE ADVANCING-LINE.
       GO TO READ-ROUTINE.
END-ROUTINE.
       CLOSE STUDENT-DATA.
       STOP RUN.
```

**Fig. 5-10** Sample COBOL program

business problems. The language which resulted was called COBOL—COmmon Business Oriented Language.

Although the COBOL language actually can be lengthy in its description of an operation, it also is easily used by people who have no knowledge of machine language. COBOL is the most "English-like" of the procedure-oriented languages.

Fig. 5-10 is an example of a COBOL program which was written to print all the names of students who received a grade point average of 3.5 or higher for the current term. Try reading it to see if you could have figured out what the program is about just from the COBOL language itself.

Reading the COBOL program above, can you see where student data are read into the computer and where the decision about GPA's is made? (Lines 4 and 5 of the program are involved in these steps.) When the end of data is recognized by the computer, processing stops. Did you see that lines 6 through 10 tell the computer how to proceed once a decision is made? COBOL is, indeed, a very English-like language, isn't it?

FORTRAN, meaning FORmula TRANslation, was developed by IBM for scientific problem-solving. FORTRAN is capable of expressing a problem in numeric formulas or algebraic expressions. It has the capacity to deal with large formulas and many variable expressions.

FORTRAN, like COBOL, is called a procedure-oriented language because it tends to group separate operations into one instruction or procedure. FORTRAN also makes use of groups of statements called subroutines which can be referred to many times at any point in the program where they are needed.

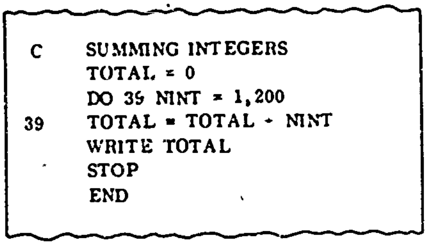
[illegible] you read over the sample FORTRAN program in Fig. 5-11, you will see that although the coding is more compact than COBOL, it is, indeed. a more mathematical language.

Could you tell that the program in Fig. 5-11 gives the instructions for summing the integers from 1 to 200? Look at the instruction in line 3: DO 39 NINT = 1,200. This is where the program tells the computer what integers are to be processed in the program (integers 1 through 200) and what instructions in the program to use (instruction 39). Instruction 39 simply tells the computer what formula to use to sum the integers. This instruction is a very simple example of a procedure.

The simplest language of all to learn is BASIC (Beginners All-purpose Symbolic Instruction Code). Developers of programming languages wanted a stepping-stone to learning more complex languages such as FORTRAN, and they created BASIC for this purpose.

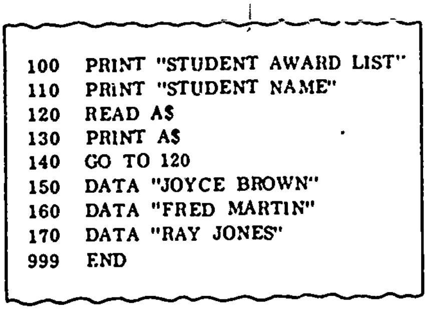
BASIC is very often used in schools to teach the fundamentals of programming techniques. Not only is it fairly easy to learn, but it can be used to program the solutions to mathematical and general problems alike. Read through the BASIC program in Fig. 5-12 to see if you can tell what kind of problem it solves.

You can see quite easily that this program tells the computer to print the student award list consisting of three names given as data. Notice that each line is separately numbered (100, 110, etc.) and that brief English command-words are used—PRINT, READ, DATA, GO



```
C       SUMMING INTEGERS
        TOTAL = 0
        DO 39 NINT = 1,200
39      TOTAL = TOTAL + NINT
        WRITE TOTAL
        STOP
        END
```

Fig. 5–11 Sample FORTRAN program



```
100  PRINT "STUDENT AWARD LIST"
110  PRINT "STUDENT NAME"
120  READ A$
130  PRINT A$
140  GO TO 120
150  DATA "JOYCE BROWN"
160  DATA "FRED MARTIN"
170  DATA "RAY JONES"
999  END
```

Fig. 5–12 Sample BASIC program

TO and END. These characteristics of BASIC help to make it an easy language to use. Now answer the check questions on this section.

## Check your understanding

1. The set of symbols the computer itself understands is called

   a. symbolic coding.
   b. assembly code.
   c. machine language.
   d. FORTRAN.

2. Most machine languages use

   a. numeric codes.
   b. alphabetical names for data.
   c. operation codes.
   d. English sentences.

3. The coding you see below represents a type of programming language called

   a. address language.
   b. assembly language.
   c. mnemonic language.
   d. high-level language.

| Operation | Address |
|---|---|
| DVH | 2700 |

4. Once assembly language is coded, it is

   a translated by the computer into machine language.
   b. translated into a high-level language.
   c. considered to be in machine language form.
   d. none of the above.

5. Procedure-oriented languages are also called

   a. higher-level languages.
   b. lower-level languages.
   c. operation languages.
   d. basic languages.

6. One high-level language instruction ultimately may be translated into

   a. only one machine language instruction.
   b. one or more machine language instructions
   c. one assembly language instruction.
   d. one or more assembly language instructions.

## "DEBUGGING" A PROGRAM

Once a program has been completely written and checked by the programmer, it is transferred onto punched cards or paper tape by a data preparation clerk. The punched cards or paper tape are always carefully checked (or "verified") for errors before being returned to the programmer. When he or she has the prepared program in hand, the programmer can begin to "debug" it—that is, to locate and correct any problems in it.

The specific methods of debugging programs vary with the types of computer and the programming language used. But there are three basic steps in the debugging procedure used in nearly all situations.

- A computer listing of the program is obtained and is thoroughly checked and corrected by the programmer at his or her desk.
- The corrected program is compiled by the computer, and the compiled program listing is gone over carefully by the programmer at his or her desk. Any and all errors are carefully corrected. This step is repeated until the compiled listing shows no errors at all.
- The final compiled version of the program is tested with sample data to determine if the program yields the correct output. If any errors are found in this step, the program is corrected again and retested until all output is correct.

We will look at these steps individually to see what techniques are involved in each one.

## Desk checking the program listing

Once the program has been "prepared" on cards or tape, the programmer usually has it run on the computer and "listed." That is, each instruction is read by the computer and is printed out or "listed." It is this listing which the programmer uses to begin debugging the program at his or her desk.

At the desk, the programmer studies the listing to identify any apparent "bugs." By reading the listing carefully and comparing it, line by line, with the original hand-written (or "coded") program, the programmer usually easily finds any clerical errors in the program, such as misspelled items, lines left out, and so forth. When all the clerical errors have been found, the programmer sends the incorrect cards or tape back to the data preparation clerk with instructions on what corrections need to be made.

As well as looking for clerical errors in the program listing, the programmer must check the logic of the program, looking for any errors in the logical sequence of steps, the internal logic in each step, and so forth. The process of looking for logical errors is, of course, more difficult than that of finding clerical mistakes. It is not a matter of simply verifying the spelling or information in each line. The programmer must follow the logic of each line to see that all processes are correctly defined and follow correctly one after the other. He or she must also keep track of (or "trace") the overall logic of the program to see, for example, that all alternatives in the program are properly defined and complete, and that the flow from beginning to end is correct. This procedure is usually called "tracing" the program.

In the process of tracing at the desk, there are several key points which the programmer may ~~use as~~ guide posts. These checkpoints can be summarized in the following questions:

1. Is all data input appropriately?
2. Is all data accurately defined?
3. Does each decision have at least two alternatives (yes, no, =, >, <)?
4. Are all processes correctly defined?
5. Are all required reports or results accurately defined?
6. Are all results output appropriately?

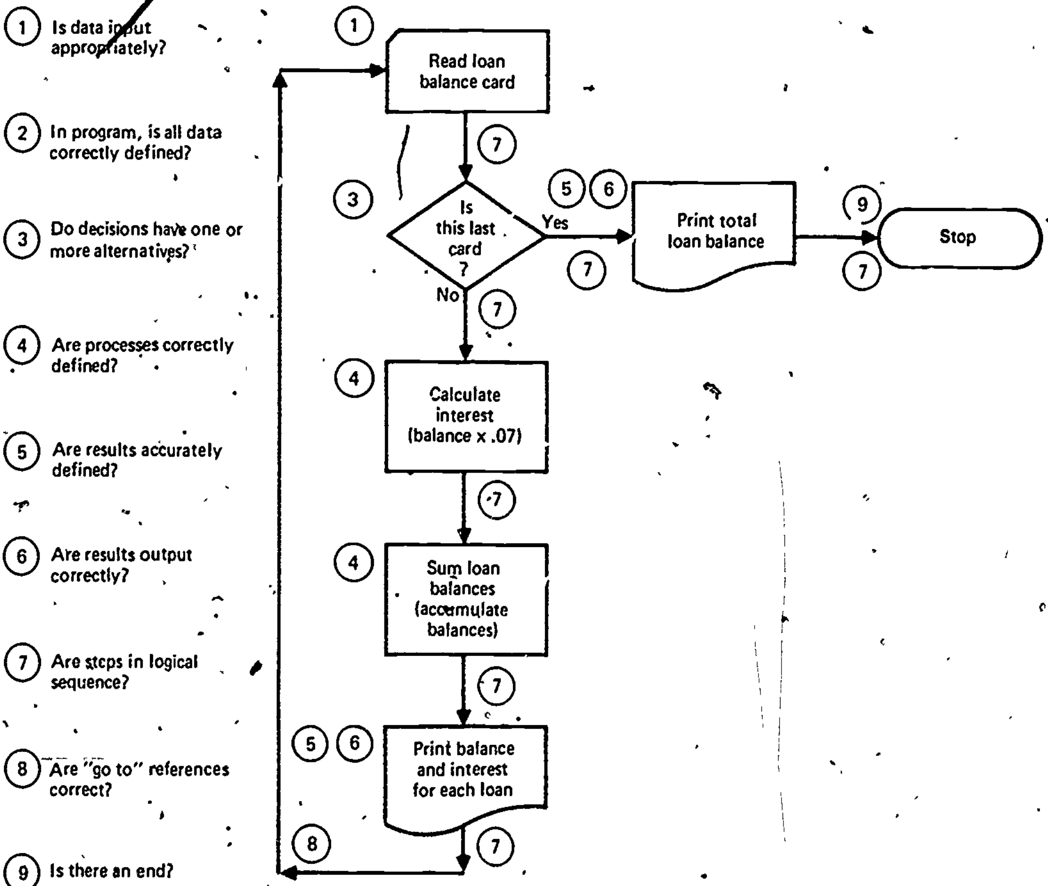
7. Does each step proceed to the next in logical sequence? (Is any step omitted? Does any step lead nowhere?)
8. Are all the "go to" directions correct?
9. Is there an end to the program?

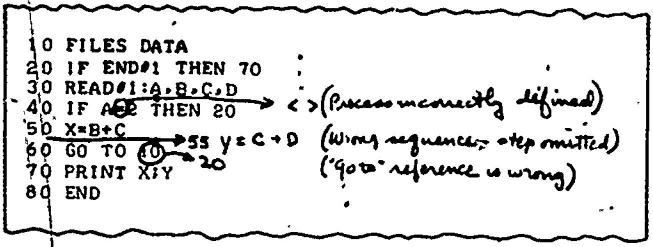
You can probably get a clear idea of how these questions help trace the key "logic" points in a program by seeing where they fit on the flowchart of a program shown in Fig. 5-13.

Fig. 5-14 is a computer listing of a BASIC program showing sample programmer's notes on the errors in it.

When the computer listing has been thoroughly desk checked and debugged and the cards or tape have been appropriately corrected, the programmer is ready to proceed to the second basic step in the debug-

**Fig. 5-13** Flowchart with key checking points


1 Is data input appropriately?
2 In program, is all data correctly defined?
3 Do decisions have one or more alternatives?
4 Are processes correctly defined?
5 Are results accurately defined?
6 Are results output correctly?
7 Are steps in logical sequence?
8 Are "go to" references correct?
9 Is there an end?
Read loan balance card
Is this last card ?
Yes
No
Print total loan balance
Stop
Calculate interest (balance x .07)
Sum loan balances (accumulate balances)
Print balance and interest for each loan



```
10 FILES DATA
20 IF END#1 THEN 70
30 READ#1:A,B,C,D
40 IF A=B THEN 20  → < > (Process incorrectly defined)
50 X=B+C           → 55 y = C + D  (Wrong sequence; step omitted)
60 GO TO 40 → 20                   ("go to" reference is wrong)
70 PRINT X;Y
80 END
```

**Fig. 5–14** Computer listing showing errors

g'ng procedure—that of desk checking the *compiled* program listing. Before studying this second step, however, use the questions below to see if you have a clear understanding of what goes into the first basic step.

## Check your understanding

1. The first step in debugging a program is best described by which of the following statements?

   a. The cards or tape are verified by the data preparation clerk.
   b. The program is run by the programmer.
   c. The hand-written (or "coded") program is debugged through desk checking.
   d. The program is "listed" on the computer, and the listing is studied to find and correct all clerical and logic errors in the program.

2. Clerical errors can be most readily found by

   a. reading the program listing and comparing it, line by line, with the original coded program.
   b. debugging the cards or tape.
   c. flowcharting the program.
   d. using the checkpoints in the tracing procedure.

3. Which of the following is a special aid to the programmer when debugging the program's logic?

   a. The checkpoints in the tracing procedure
   b. Techniques for comparing the program listing and the original coded program
   c. Flowcharting procedures
   d. None of the above

4. Which of the following are checkpoints in the tracing procedure used by programmers when desk checking their programs?

   a. Are all the flow lines connected?
   b. Does each decision have two or more alternatives?
   c. Is all data correctly input?
   d. Is the sequence from step to step logical?
   e. Are the flowcharting symbols correct?
   f. Is each item spelled correctly?
   g. Is all output accurately defined?
   h. Are all possible processes used?
   i. Are all processes defined accurately?

5. The first basic step in debugging is desk checking the program listing and flowchart. The second step is

   a. running the program on the computer to see if it works.
   b. desk checking the original coded program and flowchart for clerical and logic errors.
   c. running the program and desk checking the output to see if it is correct.
   d. compiling the program and desk checking the compiled program listing.

## Desk checking the compiled program listing

• When the corrected program is submitted foı compiling on the computer, what actually takes place is that the computer translates the program from its high-level language into machine language using its own compiler program to do the translation. In the process of compiling a high-level language program, the compiler not only translates and stores the machine-language version of the program, but it will list the high-level language version and give error messages for any instructions it cannot understand or use. You can see that such a listing with error messages on it would be a crucial debugging aid.

You may wonder why this step is not done first. There is a reason—an important one. Making a compile run on the computer uses significantly more computer time than getting a simple computer listing, and every second of computer time *costs*. It is, consequently, much more economical to do the first phase of debugging from a computer listing. Once the clerical errors and the most obvious errors in logic have been corrected, then it is worth the cost of making a compile-run to help in the final debugging of the program.

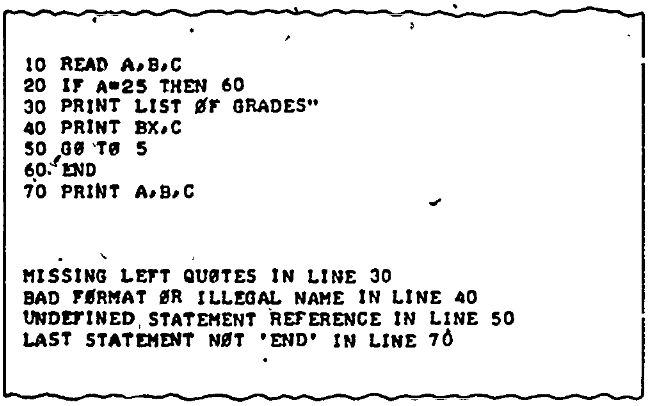
Although programming languages and error message signals differ widely, the example of a compiled program listing in Fig. 5-15 should give you an idea what such a listing looks like.

As you can see, the error messages are quite clear on this compiled listing. Notice that the compiler even picks up errors in programming conventions such as quotation marks put in incorrectly or omitted.

With the compiled listing in hand, the programmer returns to the desk to check through it carefully and to correct each problem. When the program has been debugged on the basis of the compiled listing, the programmer has the cards or tape appropriately corrected by the data preparation department. Since some of the corrections may involve errors and some corrections may affect other parts of the program, creating problems, the debugged program must be compiled again to see if there are any new errors to correct.

The process of obtaining a compiled program listing, carefully desk checking it, carefully correcting the program, and then re-compiling it goes on until the compiled listing is free of error messages.

Along with a compiled listing, some compilers are designed to print out a complete listing of locations of data for the program. This map of storage locations is often called a "memory dump." When complex or obscure errors are indicated on the compiled listing, the pro-



```
10 READ A,B,C
20 IF A=25 THEN 60
30 PRINT LIST ØF GRADES"
40 PRINT BX,C
50 GØ TØ 5
60 END
70 PRINT A,B,C


MISSING LEFT QUØTES IN LINE 30
BAD FØRMAT ØR ILLEGAL NAME IN LINE 40
UNDEFINED STATEMENT REFERENCE IN LINE 50
LAST STATEMENT NØT 'END' IN LINE 70
```

Fig. 5-15 Sample compiled program listing

grammer can often discover the exact problem by "tracing" through the data in storage as shown in the memory dump.

Since tracing through a memory dump can be a tedious chore, many computers offer the programmer a speedier way of tracing through the steps of the program. A "trace" instruction may be included in the program to be compiled which will cause the computer to list each action it takes as it goes from one step to the next through the instructions in the program. Then, studying the list of actions taken at each step, the programmer can more easily isolate and diagnose problems:

Such a "tracing" procedure is, of course, costly. It would only be used where necessary in the debugging of a program.

Once a program has been successfully compiled, it is ready to be tested to see if it does what it is supposed to do. In the time-sharing mode using the BASIC language, compiling is usually a part of the running of the program. In that case, the programmer usually proceeds directly from the desk checking of the program listing to a test run.

Check your understanding before going on.

## Check your understanding

1. The most common "debugging" aid offered by the computer is

   a. a listing of cards.
   b. a listing of error messages.
   c. a memory dump.
   d. a listing of actions taken by the computer.

2. A program is successfully compiled when

   a. it is coded.
   b. it is punched on some input medium.
   c. no more error messages are printed.
   d. it can be traced.

3. A programmer can trace data in storage by using

   a. a memory dump.
   b. a program listing.
   c. error messages.
   d. a template.

4. Tracing through a program can be done automatically on the computer, when the computer is designed to accept which of the following?

   a. A "trace" instruction
   b. A memory dump
   c. A "test" command
   d. A "compile" instruction

5. List the following debugging steps in the order that they occur.

   The program is compiled.
   Clerical and logical errors are corrected.
   The program is listed.
   The error messages are checked and the program is corrected.
   The program is re-compiled.

## Testing a program on the computer

Suppose that a computer program has been carefully flowcharted and coded, and that the computer programmer has skillfully taken care of any clerical or logical errors so that a "clean" or error-free compiled program listing has been received. The program is now ready to run on the computer. But do you know yet whether or not it will run correctly? Have you seen the output? Not yet. And, it is the resulting output of a program that proves the program's reliability in solving a problem.

In spite of the very careful debugging a program goes through, very few debugged programs run perfectly the first time. In fact, no one even expects them to. The reason is very simple. As careful as you may be in coding and "debugging" your program, something may have been overlooked. Making a "test run" of the program will show any "things" which have been overlooked.

As you saw in the discussion of desk checking, a programmer checks every possible path in the program logic. A test run should, in effect, do the same thing. Rather than using a complete set of data to test a program, however, the programmer will often use selected sets of data, In this way the parts of the program that will be tested can be controlled. Using large amounts of data which test the same logical path can be eliminated. Data used in a test run are also selected to exercise the extremes of data to be processed by the program. Where possible, the programmer will check boundary value situations against known results to test the accuracy of the output.

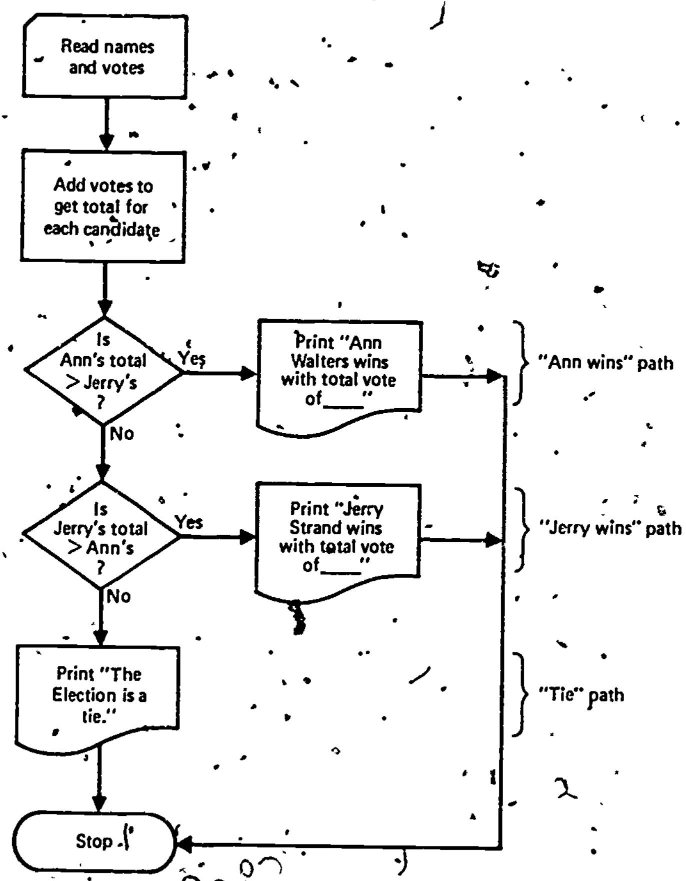
A flowchart usually clearly outlines the various alternative paths present in a program. On the sample flowchart in Fig. 5-16, each of the alternative paths has been marked so you can see the possible logical paths which might be tested separately.

As you can see, there are three logical paths to be tested in this program:

1. The path if Ann wins.
2. The path if Jerry wins.
3. The path if there is a tie.

In many cases, the actual data to be processed by a program like this may consist of 1,000 cards (representing the votes). Obviously, a

Fig. 5-16 Flowchart of a program to be tested


Read names and votes
Add votes to get total for each candidate
Is Ann's total > Jerry's ?
Yes
No
Print "Ann Walters wins with total vote of____"
"Ann wins" path
Is Jerry's total > Ann's ?
Yes
No
Print "Jerry Strand wins with total vote of____"
"Jerry wins" path
Print "The Election is a tie."
"Tie" path
Stop

deck of 1,000 cards comprises a large volume of data to run through an untested program. Computer time is wasted if the results of this large a test run are incorrect. Only a few cards are needed as input to a test run as long as the cards used test each alternative path in the program logic.

Let's look at the selected data you would need to test the program illustrated in the flowchart in Fig. 5-16. Obviously, in this case, data comes into the program on punched cards. We can look at just what data will make up a test input deck.

Since the program calls for tallying votes for Ann Walters and votes for Jerry Strand, the programmer would probably need at least two cards registering votes for each candidate to see if the program adds correctly.

A test run with these four cards would not only test the addition of the program, but it would test whether or not a correct "tie" report were produced.

To test the "Ann wins" path next, the programmer could just add another card registering a vote for Ann to the original deck of four and make another test run.

Finally, to test the "Jerry wins" path, two more cards registering votes for Jerry could be added to the deck and a third test run made.

By running all the test data through a program on the computer, a programmer can see if each selected card is processed correctly. If any portion of the final test output is incorrect, the programmer must correct the program, recompile it, and submit another test run. When the results of the test run are accurate, the testing phase is completed. At this point, the programmer is ready to document and then submit the program for use. Before learning to document your program, check your understanding to this point.

## Check your understanding

1. Once a programmer receives a "clean" compiled program,

   a. the program can be submitted for use.
   b. a test run of the program must be performed.
   c. the program is completed.
   d. all of the above are correct answers.

2. Which of the following proves the reliability of a computer program?

   a. An error-free compiled program listing
   b. The resulting output
   c. The input
   d. The memory dump

3. A test run is used specifically to check

   a. for clerical errors in the program.
   b. for compiling pıoblems.
   c. every path in the program logic and the resulting output.
   d. all of the above.

4. Test data usually consists of

   a. all the actual data to be used by the program.
   b. all punched cards.
   c. selected data which tests a few logical paths in the program.
   d. none of the above.

5. If the results of a test run are incorrect, the programmer will

   a. correct the program and submit another test run.
   b. turn the program over for use.
   c. recode the entire program from scratcn.
   d. change the test data deck.

## DOCUMENTING A PROGRAM

The people who work in a computer center often have to rely on each other for information. The systems analyst, who designs overall computer systems to meet customers' computer needs, relies on the customers for a definition of their needs. The programmer relies on the systems analyst for accurate specifications for the programs required by the system. The computer operator must rely on the programmer for instructions on how to run the programs he or she has written. Obviously, these people must communicate with each other. Because computer centers often have changes in personnel or assignments, communication is most effective in a written form. All the written communication about a computer job is called documentation. Each member of the computer center staff makes his or her own contribution to the written documentation used in the center.

Often documentation standards are created by the collective staff according to the needs of each department and are approved by the computer center manager. In this section you will see some steps the programmer follows in documenting computer programs.

The computer programmer is generally responsible for two kinds of documentation: (1) general information about a program for the computer center records and (2) specific information to the computer operator on how to run the program.

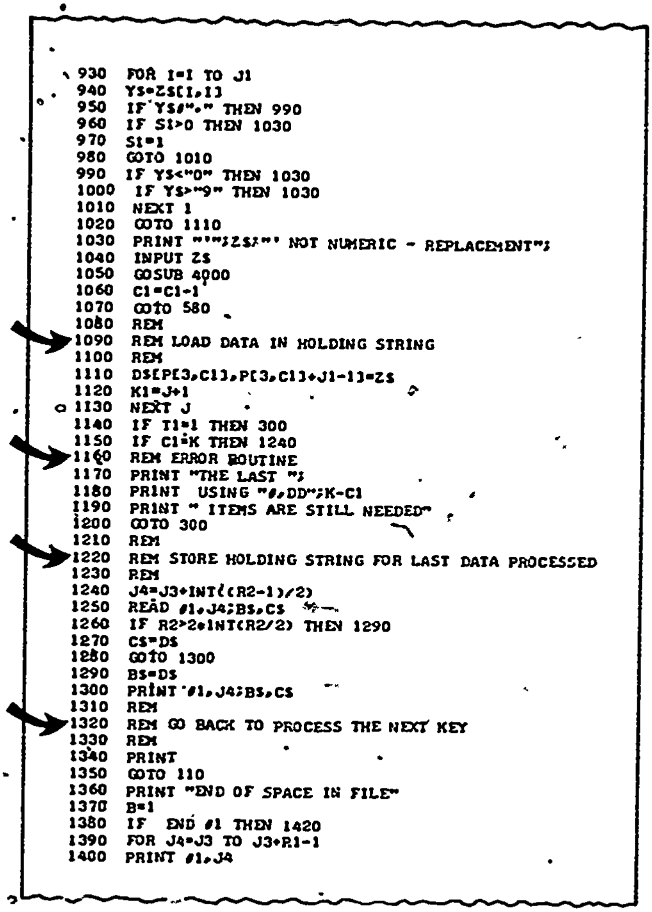
## General documentation

A general documentation for a program includes the final flowchart, a program listing, a program description, and description of input and output. Let's look at each in turn.

You saw in the section about "debugging" that the programmer may have to make changes in a program which change the original flowchart. The programmer must keep track of all such changes so that a flowchart can be drawn showing the final solution to the problem. As with any documenting process, the programmer will usually follow standards set by the computer center staff. She or he may need to use a template with specific symbols of a particular size. The programmer may also have a standard flowchart paper to use as the flowchart may be placed in a binder with other information. Whatever the flowchart conventions for a particular computer center, a programmer must supply a final flowchart of the program for general use.

Once the program has been debugged and successfully tested, a program listing is obtained for use in documentation. In addition, it is helpful in the documentation to have comments inserted in the program before it is listed as guides to any new programmer who may have to update the program. Nearly every programming language supplies some means of making remarks in a program. The sample program in Fig. 5-17 shows one way remarks can be placed in a program. Look at the notes pointed out by the arrows.

Often computer programmers write out a complete description of the program as a part of the general documentation. The general description may include an explanation of the steps in processing data, the form of input and output used, and a description of any codes used along with the meaning of each code.



```
930  FOR I=I TO J1
940  Y$=Z$[I,I]
950  IF Y$#"." THEN 990
960  IF S1>0 THEN 1030
970  S1=1
980  GOTO 1010
990  IF Y$<"0" THEN 1030
1000  IF Y$>"9" THEN 1030
1010  NEXT I
1020  GOTO 1110
1030  PRINT "'";Z$;"' NOT NUMERIC - REPLACEMENT";
1040  INPUT Z$
1050  GOSUB 4000
1060  C1=C1-1
1070  GOTO 580
1080  REM
1090  REM LOAD DATA IN HOLDING STRING
1100  REM
1110  D$[P[3,C1],P[3,C1]+J1-1]=Z$
1120  K1=J+1
1130  NEXT J
1140  IF T1=1 THEN 300
1150  IF C1=K THEN 1240
1160  REM ERROR ROUTINE
1170  PRINT "THE LAST ";
1180  PRINT  USING "#,DD";K-C1
1190  PRINT " ITEMS ARE STILL NEEDED"
1200  GOTO 300
1210  REM
1220  REM STORE HOLDING STRING FOR LAST DATA PROCESSED
1230  REM
1240  J4=J3+INT((R2-1)/2)
1250  READ #1,J4;B$,C$
1260  IF R2>2*INT(R2/2) THEN 1290
1270  C$=D$
1280  GOTO 1300
1290  B$=D$
1300  PRINT #1,J4;B$,C$
1310  REM
1320  REM GO BACK TO PROCESS THE NEXT KEY
1330  REM
1340  PRINT
1350  GOTO 110
1360  PRINT "END OF SPACE IN FILE"
1370  B=1
1380  IF  END #1 THEN 1420
1390  FOR J4=J3 TO J3+R1-1
1400  PRINT #1,J4
```

**Fig. 5–17** Sample documentation listing with comments

Finally, general documentation usually includes information on the input and output formats used with this program. Special diagram forms are commonly used for this purpose to show just how input must be arranged on the media or how the output forms should look. Fig. 5-18 gives an example of a card layout form which shows just how input must be prepared on cards for use with a specific program.

As you can see, data is grouped on the input card in a certain format. The card layout sheet gives general information about the input. Notice that special processing codes and their meanings are also listed on the card layout sheet. This information about input is helpful to the keypunch operator who punches card input. It is also helpful to other computer center staff members who need to understand how a computer program works.

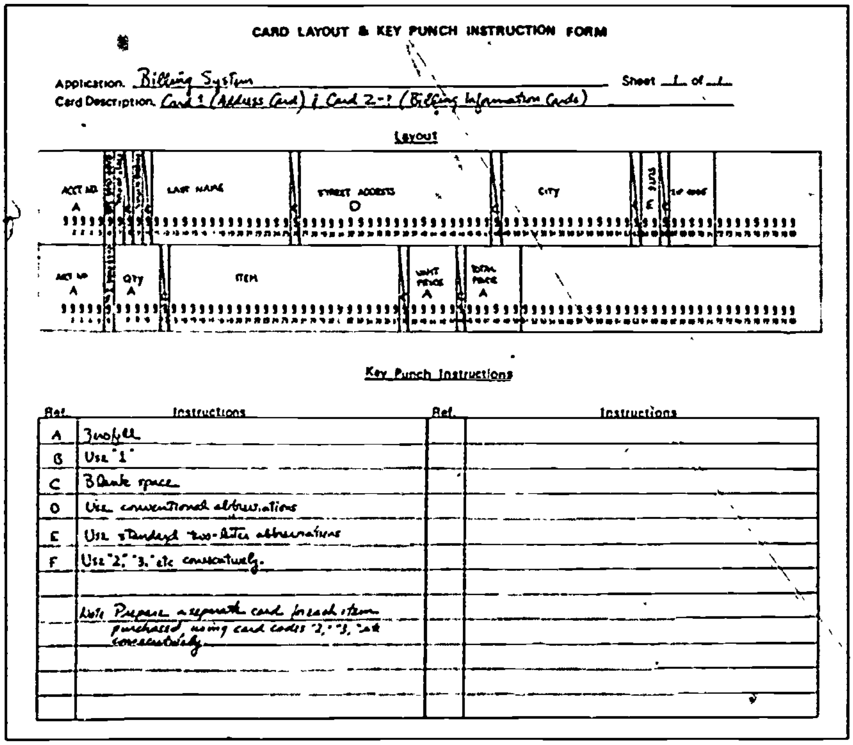
Output of a program is also documented with a layout sheet for the particular output medium used. Printer reports, for example, are drawn up on a print layout sheet like the one you see in Fig. 5-19.

As you can see on the sample print chart, the exact print positions of data on a printed report are described. Often a preliminary-print layout is created by the programmers before they code a program. During the testing phase, changes to the report format may have to be made. So programmers are always prepared to correct the print layout sheets as they go along. The final layout sheet submitted as documentation reflects the finally tested print format.

### Specific documentation

In the specific documentation, the programmer is responsible for telling the computer operator how to run the programs. Often documentation for operating procedures is also standardized by the computer center staff and approved by the computer center manager. An example of instructions for documenting a program for the operations staff in a typical computer center is given in Fig. 5-20.

Fig. 5–18 Sample card layout



CARD LAYOUT & KEY PUNCH INSTRUCTION FORM

Application. Billing System Sheet 1 of 1

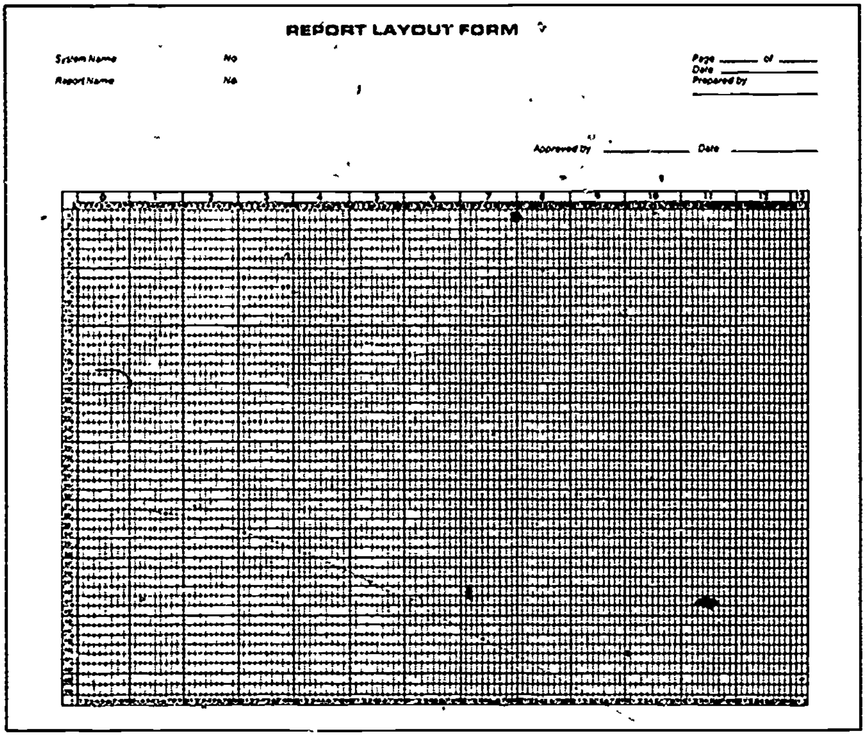
Card Description. Card 1 (Address Card) & Card 2-? (Billing Information Cards)

Layout

ACCT NO
A
LAST NAME
STREET ADDRESS
D
CITY
STATE
E
ZIP CODE
ACCT NO
A
QTY
A
ITEM
UNIT PRICE
A
TOTAL PRICE
A

Key Punch Instructions

| Ref. | Instructions | Ref. | Instructions |
|---|---|---|---|
| A | Zero fill | | |
| B | Use "1" | | |
| C | Blank space | | |
| D | Use conventional abbreviations | | |
| E | Use standard two-letter abbreviations | | |
| F | Use "2," "3," etc consecutively. | | |
| | | | |
| | Note Prepare a separate card for each item purchased, using card codes "2," "3," etc consecutively. | | |
| | | | |
| | | | |
| | | | |



REPORT LAYOUT FORM

System Name  No

Report Name  No

Page ______ of ______

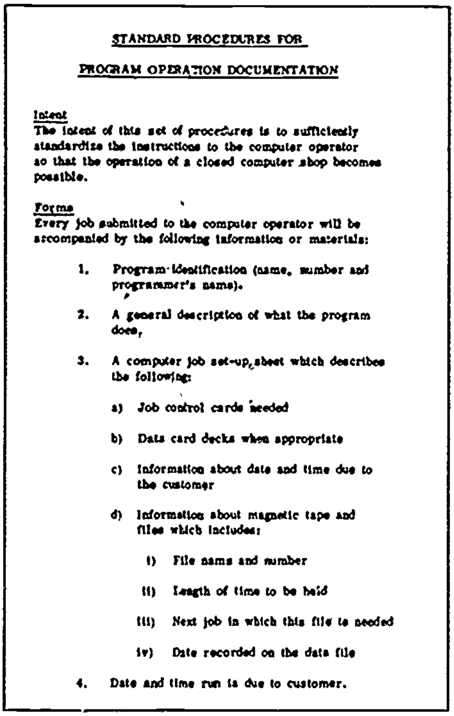
Date ______

Prepared by ______

Approved by ______ Date ______

0
1
2
3
4
5
6
7
8
9
10
11
12
13

Fig. 5–19 Sample print chart for documentation



STANDARD PROCEDURES FOR

PROGRAM OPERATION DOCUMENTATION

Intent
The intent of this set of procedures is to sufficiently standardize the instructions to the computer operator so that the operation of a closed computer shop becomes possible.

Forms
Every job submitted to the computer operator will be accompanied by the following information or materials:

1. Program identification (name, number and programmer's name).

2. A general description of what the program does,

3. A computer job set-up sheet which describes the following:

   a) Job control cards needed

   b) Data card decks when appropriate

   c) Information about date and time due to the customer

   d) Information about magnetic tape and files which includes:

      i) File name and number

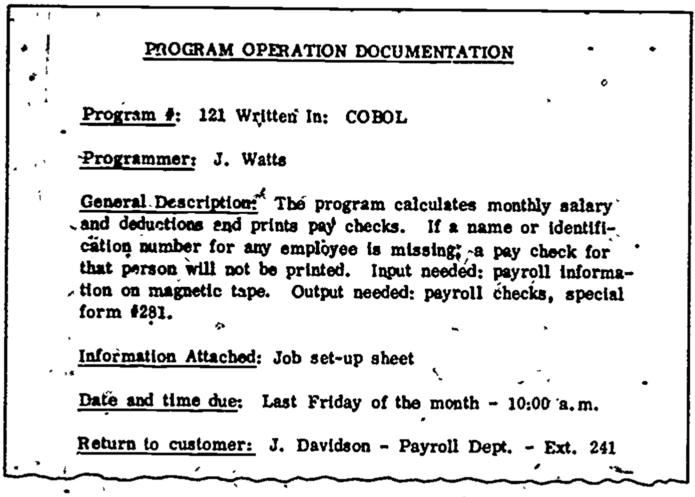
      ii) Length of time to be held

      iii) Next job in which this file is needed

      iv) Date recorded on the data file

4. Date and time run is due to customer.

Fig. 5–20 Sample documentation instruction sheet



PROGRAM OPERATION DOCUMENTATION

Program #: 121 Written In: COBOL

Programmer: J. Watts

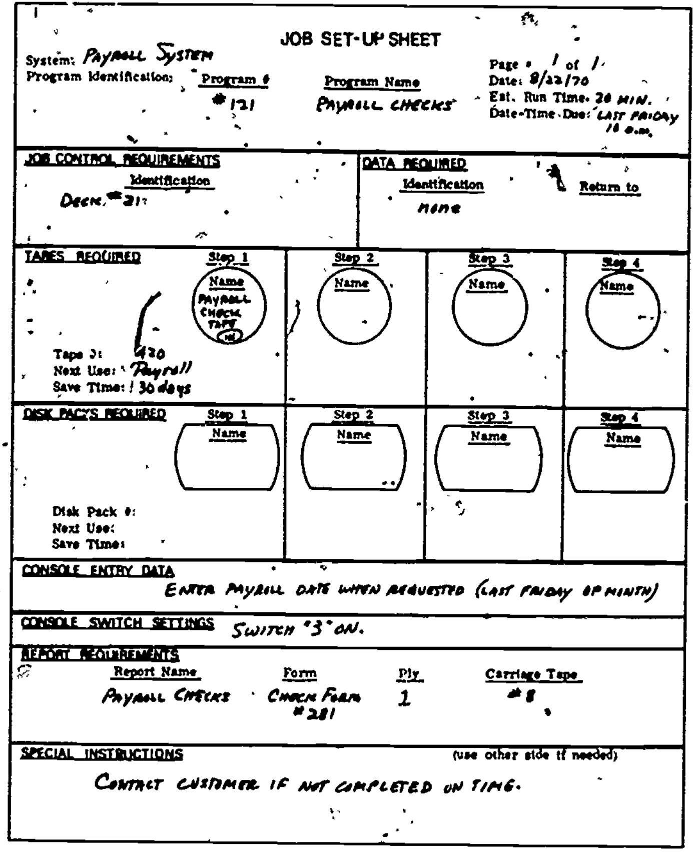
General Description: The program calculates monthly salary and deductions and prints pay checks. If a name or identification number for any employee is missing, a pay check for that person will not be printed. Input needed: payroll information on magnetic tape. Output needed: payroll checks, special form #281.

Information Attached: Job set-up sheet

Date and time due: Last Friday of the month - 10:00 a.m.

Return to customer: J. Davidson - Payroll Dept. - Ext. 241

Fig. 5-21 Sample program operation documentation sheet



JOB SET-UP SHEET

System: PAYROLL SYSTEM

Program Identification: Program # #121 Program Name PAYROLL CHECKS

Page # 1 of 1

Date: 8/22/70

Est. Run Time: 20 MIN.

Date-Time Due: LAST FRIDAY 10 a.m.

JOB CONTROL REQUIREMENTS

Identification

DECK #21

DATA REQUIRED

Identification: NONE

Return to

TAPES REQUIRED

Step 1 Name: PAYROLL CHECK TAPE

Step 2 Name

Step 3 Name

Step 4 Name

Tape #: 420

Next Use: Payroll

Save Time: 30 days

DISK PACKS REQUIRED

Step 1 Name

Step 2 Name

Step 3 Name

Step 4 Name

Disk Pack #:

Next Use:

Save Time:

CONSOLE ENTRY DATA

ENTER PAYROLL DATE WHEN REQUESTED (LAST FRIDAY OF MONTH)

CONSOLE SWITCH SETTINGS SWITCH "3" ON.

REPORT REQUIREMENTS

| Report Name | Form | Ply | Carriage Tape |
|---|---|---|---|
| PAYROLL CHECKS | CHECK FORM #281 | 1 | #8 |

SPECIAL INSTRUCTIONS (use other side if needed)

CONTACT CUSTOMER IF NOT COMPLETED ON TIME.

Fig. 5-22 Sample job set-up sheet

As you can see in this example, a programmer is usually required to submit rather detailed information about running the programs. A computer operator is always concerned about efficient use of computer time. The easiest way to minimize the time it takes to run a program is to have very well-written instructions available.

Fig. 5-21 is an example of a partial program operation description which follows the instructions given in Fig. 5-20.

As is indicated on the program operation documentation form in Fig. 5-21, a job set-up sheet is to be found attached. An example of such a sheet is given in Fig. 5-22.

Look the example over carefully. You can see that the program identification is complete, including what card decks are needed and what output forms are needed when running the program. Also, if the operator must enter anything on the computer console during the run, the programmer makes a note of it on the job set-up sheet. In addition, any special computer switch settings needed are also noted.

See if general and special documentation are clear to you now by answering the check questions.

## Check your understanding

1. Program documentation is

   a. the program listing.
   b. an input description.
   c. all written communications about a program.
   d. instructions on programming.

2. Which of the following is true about documentation standards?

   a. Standards are created by the computer center staff according to the needs of each department.
   b. Standards are usually approved by the computer center manager.
   c. Both of the above are true.
   d. Neither of the above are true.

3. Which of the following kinds of documentation is a programmer responsible for?

a. General information on programs for computer center records
b. Daily operation schedule
c. Specific information on how to run a program
d. Systems analysis
e. Management reports

4. Which of the following kinds of general information are usually required for documentation from the programmer?

a. A program description
b. A systems specification
c. A program listing
d. An operating manual
e. A final flowchart
f. Input and output descriptions

5. Input or output descriptions are most often recorded on

a. punched cards.
b. paper tape.
c. layout sheets.
d. program listings.

6. Which of the following is the most important documentation to the computer operator?

a. A job set-up sheet
b. The name and location of the customer
c. Input or output layout sheets
d. A general description of the program

7. Study the job set-up sheet in Fig. 5-22 on page 202 and answer the questions below.

a. How is the program identified? What is the program number? What is the program name?
b. What magnetic tapes or disks are used for this program?
c. Describe the output form to be used with this program.

## REFERENCE MANUALS

Very few programmers, unless they have a "photographic" memory, can remember every detail about the computer system they are working with or all the possible instructions in the symbolic language they are using. Consequently, programmers make considerable use of reference manuals which give the needed details on how a system operates, what its software does, and how programming languages are used with the system.

With experience, of course, a programmer can begin to use a computer system or a language with greater ease. But, as computer programmers must eventually know about more than one programming language and more than one computer, they need to rely more and more on reference manuals and guides in their work. Not only do programmers ultimately work with several different programming languages, but each different system may require slight modifications in the way the programming language is used.

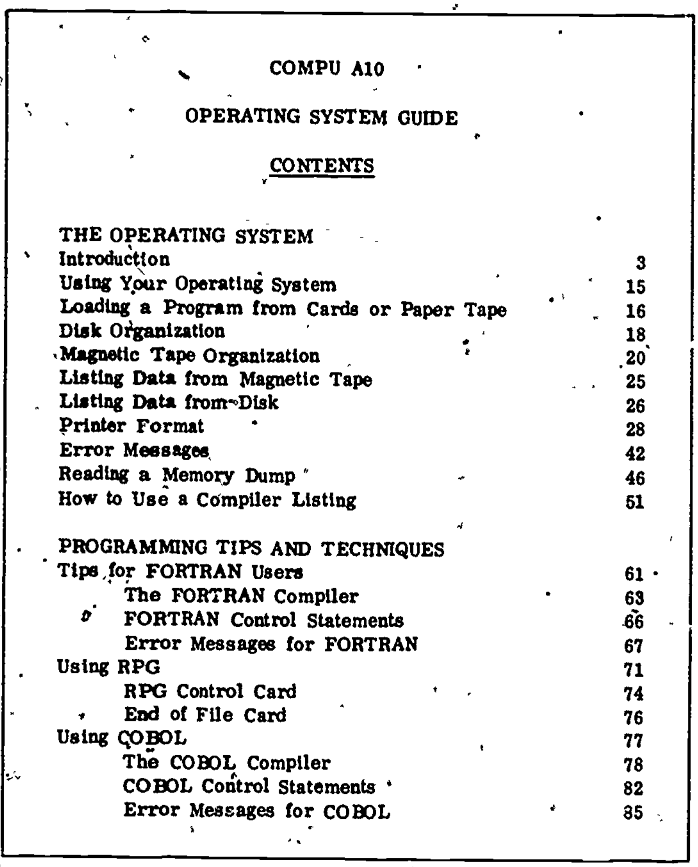
### Operating system guides

All manufacturers of computer systems provide operating system guides. They explain what hardware is available in the system as well as the details about the software (or operating system programs) designed for it. Since the information in a reference manual covers all details about the computer system, programmers don't even need to try to memorize them all. They only need to know how to use the guide so that whenever there is a question about how the computer works, they can refer to the operating system manual and find the answer.

An operating system manual usually discusses many topics. These include how the system works in general and how to use its input/output capabilities. A manual also includes whatever a programmer needs to know about how a computer's hardware accepts and processes data or programs.

In addition, the operating system manual often has general information about programming languages. Since most computer systems are capable of translating two or three symbolic languages, the operating system guide will often give a general description of how each symbolic language is used in relation to the hardware available on a particular computer.

You can see from the sample table of contents in Fig. 5-23 how an operating system manual may be organized. Read the contents carefully and notice how information about the system's input/output capabilities is entered. Also, look over the entries on programming



COMPU A10

OPERATING SYSTEM GUIDE

CONTENTS

| | |
|---|---|
| THE OPERATING SYSTEM | |
| Introduction | 3 |
| Using Your Operating System | 15 |
| Loading a Program from Cards or Paper Tape | 16 |
| Disk Organization | 18 |
| Magnetic Tape Organization | 20 |
| Listing Data from Magnetic Tape | 25 |
| Listing Data from Disk | 26 |
| Printer Format | 28 |
| Error Messages | 42 |
| Reading a Memory Dump | 46 |
| How to Use a Compiler Listing | 51 |
| PROGRAMMING TIPS AND TECHNIQUES | |
| Tips for FORTRAN Users | 61 |
| The FORTRAN Compiler | 63 |
| FORTRAN Control Statements | 66 |
| Error Messages for FORTRAN | 67 |
| Using RPG | 71 |
| RPG Control Card | 74 |
| End of File Card | 76 |
| Using COBOL | 77 |
| The COBOL Compiler | 78 |
| COBOL Control Statements | 82 |
| Error Messages for COBOL | 85 |

**Fig. 5–23** Sample table of contents for an operating system guide

tips to see what kinds of information the manual provides about programming for this computer system.

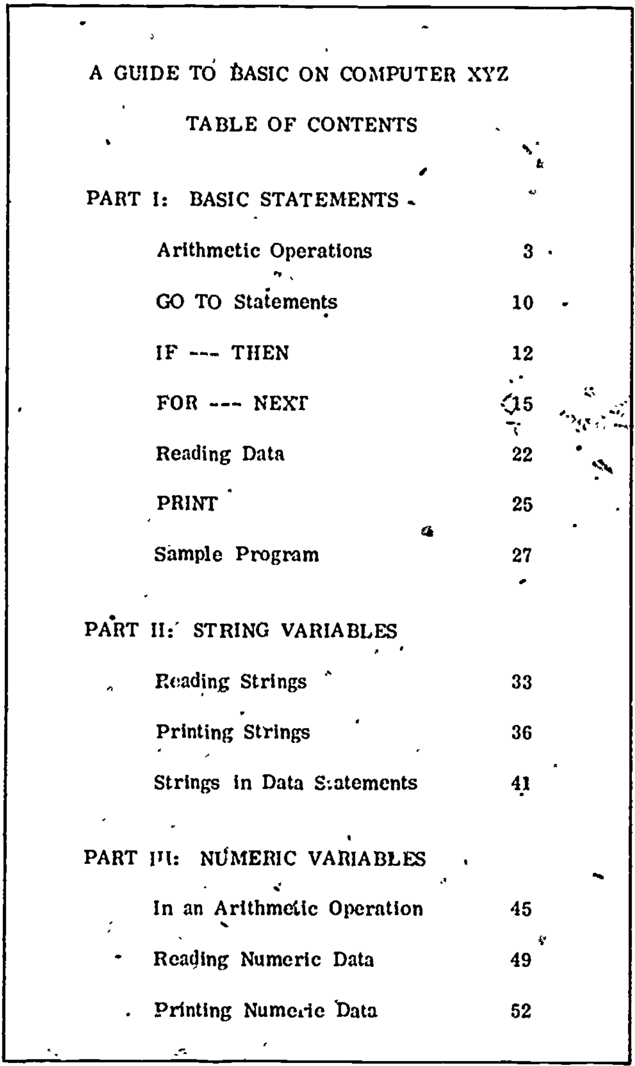
As you can see from this example, most of the programming tips and techniques found in an operating system guide pertain to programming in relation to the computer software. For example, the compiler and control statements for each language are discussed in this section. Most of the detailed elements about using a programming language, such as how to use the arithmetic operations in the language, are left out of the operating system manual. Most manufacturers find it more practical to write separate guides to the programming languages to be used on their system.

### Programming language guides

A manufacturer's programming language guide shows all the details of how each programming language must be used with the particular computer system. To ensure accurate and efficient programming for a particular system, therefore, programmers always follow the system's appropriate programming language guide carefully. The sample table of contents shown in Fig. 5-24 illustrates how complete the typical programming language guide is. Read it carefully.

You can see that Part I describes each of the several BASIC statements which may be used with this system. Parts II and III deal in detail with how variables in BASIC must be handled.

Once programmers have learned a standard language, they can then program for any system in that language with the help of the



A GUIDE TO BASIC ON COMPUTER XYZ

TABLE OF CONTENTS

PART I: BASIC STATEMENTS

| | |
|---|---|
| Arithmetic Operations | 3 |
| GO TO Statements | 10 |
| IF --- THEN | 12 |
| FOR --- NEXT | 15 |
| Reading Data | 22 |
| PRINT | 25 |
| Sample Program | 27 |

PART II: STRING VARIABLES

| | |
|---|---|
| Reading Strings | 33 |
| Printing Strings | 36 |
| Strings in Data Statements | 41 |

PART III: NUMERIC VARIABLES

| | |
|---|---|
| In an Arithmetic Operation | 45 |
| Reading Numeric Data | 49 |
| Printing Numeric Data | 52 |

**Fig. 5-24** Sample table of contents for a programming language manual

system's accompanying language guide. Often a programming language guide for a given system will reflect only minor alterations in the standard conventions of the language. On the other hand, some systems require extensive modifications in the language's use. In both cases, the manufacturer's language guides provide all the information the programmer needs to use programming languages correctly and effectively with the system.

## Subroutine libraries

In writing programs, programmers often need to instruct computers to do very routine tasks. These may include sorting data in alphabetic or numeric order, which involves long complex sets of instructions. If programmers had to write out the extensive instructions every time the routine task had to be performed, a great deal of time would be wasted.

To save programmers this time, computer manufacturers normally provide their systems with a whole set of pre-prepared programs telling the computer how to perform these routine tasks. Each of these programs is called a subroutine.

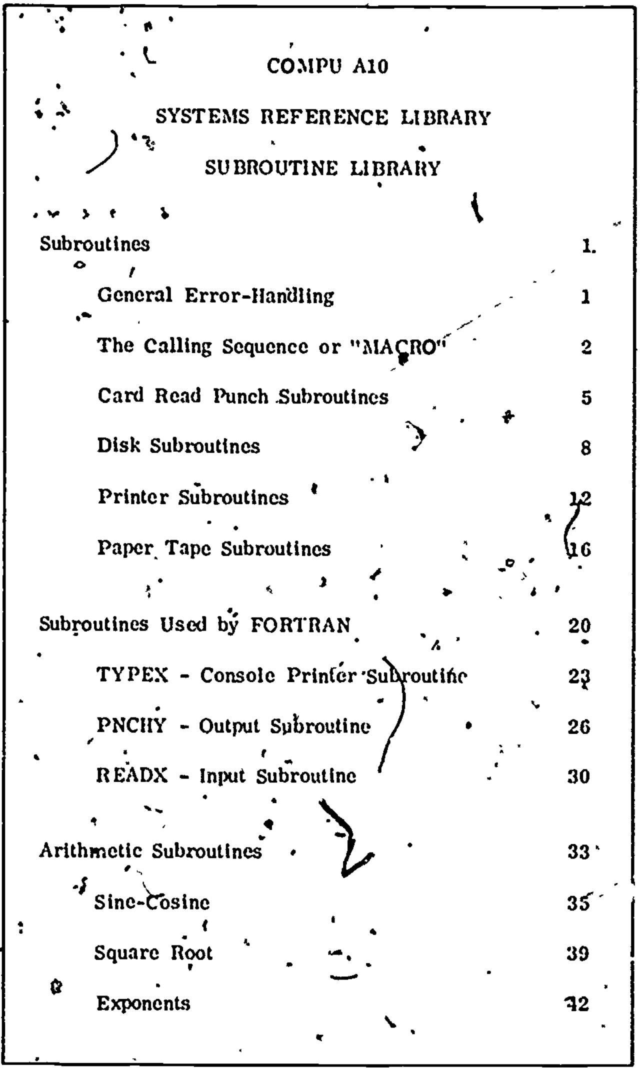
Subroutines are stored as part of a computer's operating system. Then, when the programmer comes to a place in the program where a routine task is needed for which a subroutine is available, he or she can write in a "request" or "macro" instruction to use the subroutine at that place.

The complete set of subroutines in a system is usually called the "subroutine library." Manufacturers always provide subroutine library manuals for their systems. They describe all the subroutines available in the library and give directions to the programmer on how to use them.

A table of contents for a typical subroutine library manual is shown in Fig. 5-25. Notice the different kinds of subroutines that are included in this sample library.

Can you see how computer programmers depend a great deal on reference books? Programmers can improve their knowledge of their jobs quite a bit just by learning which manual will tell them what they need to know.

Before you go on to the next section on learning and writing programs, see if you understand the use of reference manuals in a programmer's daily work by answering the following questions.



COMPU A10

SYSTEMS REFERENCE LIBRARY

SUBROUTINE LIBRARY

| | |
|---|---|
| Subroutines | 1. |
| General Error-Handling | 1 |
| The Calling Sequence or "MACRO" | 2 |
| Card Read Punch Subroutines | 5 |
| Disk Subroutines | 8 |
| Printer Subroutines | 12 |
| Paper Tape Subroutines | 16 |
| Subroutines Used by FORTRAN | 20 |
| TYPEX - Console Printer Subroutine | 23 |
| PNCHY - Output Subroutine | 26 |
| READX - Input Subroutine | 30 |
| Arithmetic Subroutines | 33 |
| Sine-Cosine | 35 |
| Square Root | 39 |
| Exponents | 42 |

Fig. 5–25 Sample table of contents for a subroutine library manual

# Check your understanding

1. Programmers make considerable use of reference books because

   a. the details of programming often vary from system to system.
   b. the standard conventions of a language are hard to learn.
   c. they usually have to program in too many different languages to learn them.
   d. all of the above are true.

2. Information about how the system operates can usually be found in

   a. the system's programming language manual.
   b. the systems operating system guide.
   c. a BASIC manual.

3. To learn how to use FORTRAN most effectively with a specific computer system, the programmer would refer to

   a. a standard FORTRAN language textbook.
   b. any system's FORTRAN language guide.
   c. the specific system's operating system manual.
   d. none of the above.

4. A subroutine library manual

   a. describes the available subroutines in a computer system.
   b. tells where in the computer center's library to find books on subroutines.
   c. tells how to use the system's subroutines.
   d. does none of the above.

5. Assume you are programming for the Compu A10 system, and that you have the operating system guide whose table of contents is shown in Fig. 5-23 on page 206. Look at the table of contents carefully. What pages would you look on to find out about how to use the Compu A10 operating system, how to read a Compu A10 memory dump, what the Compu A10 COBOL compiler will list, the format of the printer, COBOL error messages?

6. If you were programming in BASIC on Computer XYZ's system, you would have the system's GUIDE TO BASIC available. The table of contents for this guide is shown on page 207. Which of the following topics would you find discussed in this guide?

   a. How to call on the Computer XYZ's alphabetizing subroutine
   b. How to program in BASIC for Computer XYZ
   c. How to use BASIC statements with XYZ
   d. How to program for XYZ in FORTRAN
   e. How XYZ reads string variables in BASIC
   f. How to use XYZ's line printer

7. Suppose you were programming for the same system and you were looking at its subroutine library manual's table of contents shown on page 209. On what pages would you find details on the macro-instructions needed to call on subroutines, printer subroutine, FORTRAN input subroutine, and alphabetizing subroutines?

## LEARNING BASIC AND WRITING PROGRAMS

### The characteristics of a program

As you already know, BASIC is probably the easiest high-level programming language to learn. In this section, you will apply yourself to learning the fundamentals of programming BASIC.

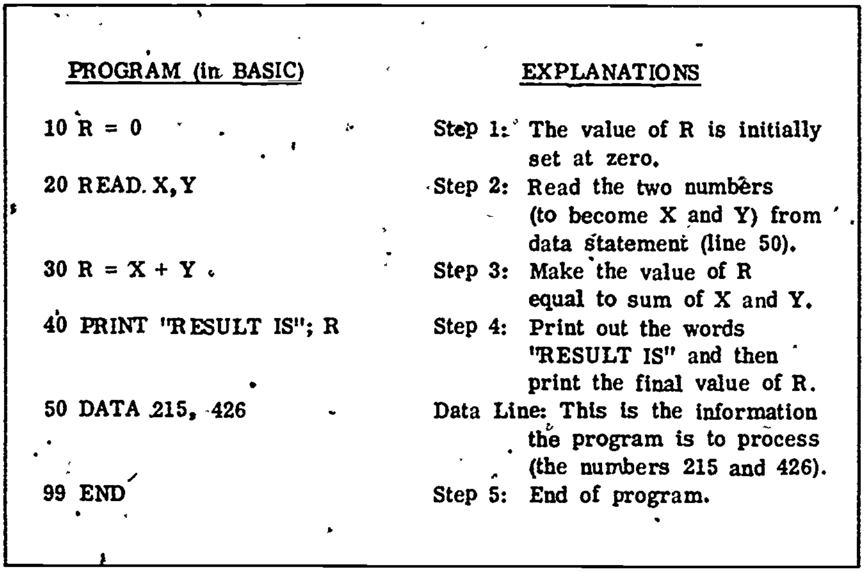
Before beginning your study of BASIC, however, you should be sure that you have a general understanding of just what a computer program is.

A computer program in any programming language is a complete set of instructions which tells a computer, step-by-step, how to process certain information (input) to obtain and print out certain other information (output).

As an example of a simple program, look at the BASIC program in Fig. 5-26 with the line-by-line explanations. Can you tell what this program is designed to do?

From the explanations in Fig. 5-26, you should have gotten the idea that the program instructs the computer in processing two numbers (215 and 426) to obtain and print out their sum.

This sample program illustrates several important features of programs in general. It should be clear that the program is written in a precise way, using key words and special punctuation rules. You can also see that the instructions follow one after the other in close, logical order. These are characteristics of all high-level languages. Each high-level language has its own rules for setting up a program, its specific instruction words and punctuation conventions, and its own rules for constructing sets of instructions to accomplish certain things.



| PROGRAM (in BASIC) | EXPLANATIONS |
|---|---|
| 10 R = 0 | Step 1: The value of R is initially set at zero. |
| 20 READ. X,Y | Step 2: Read the two numbers (to become X and Y) from data statement (line 50). |
| 30 R = X + Y | Step 3: Make the value of R equal to sum of X and Y. |
| 40 PRINT "RESULT IS"; R | Step 4: Print out the words "RESULT IS" and then print the final value of R. |
| 50 DATA 215, 426 | Data Line: This is the information the program is to process (the numbers 215 and 426). |
| 99 END | Step 5: End of program. |

**Fig. 5–26** BASIC program for obtaining the sum of 215 and 426

The example also shows some of the particular characteristics of the BASIC language. First, notice that the key words and symbols in the instructions give a good clue about what the instruction involves. For example, the second line ("READ X, Y") clearly suggests that some numbers are to be read in or input, and the third line ("R = X + Y") clearly indicates that the two numbers are to be added together. This characteristic of the BASIC language adds to the ease of learning BASIC.

Notice how concise each line of instruction is and that each line is numbered. This conciseness makes BASIC relatively quick to write and to input into a computer. The line numbers not only help the programmer keep the instructions in order, but serve as a quick reference to individual lines of instruction.

Programs in BASIC can, of course, become much more complex. However, from the simple example above you should have a good idea of what a BASIC program looks like and how it may instruct a computer in processing certain information to output certain results.

Now check your understanding.

# Check your understanding

1. Which of the following best describes a program?

   a. A set of data put in logical order
   b. A set of instructions for entering data
   c. A series of print instructions
   d. A set of instructions for inputting, processing, and outputting data

2. Of the following, which is true of high-level programming languages?

   a. They are written using special words according to precise rules and conventions.
   b. They use the punctuation conventions normally used in everyday English.
   c. They are code languages which vary from programmer to programmer.
   d. All of the above are true.

3. Which of the following are characteristics of the BASIC programming language?

   a. Code instruction words and symbols are difficult to learn to read.
   b. Each instruction line is numbered.
   c. The language is concise.
   d. Key instruction words and symbols give good clues about what the instructions involve.
   e. The language uses mostly complete English sentences,

## Learning to program in a programming language

Now you are ready to study a programming language in detail. Before going on, ask your instructor to recommend a manual which can teach you the elements of programming in BASIC. If your computer system excludes the BASIC language, of course, your instructor may recommend a manual for studying some other language, such as FORTRAN, PL/1, or COBOL.

## Programming: two kinds

The kind of programming you have been learning about is called *applications* programming. It tells the computer what to do with certain data to produce certain results. This kind of programming is required whenever a problem needs solving and a computer can be "applied" in solving it. The problem might be "to compute and print paychecks for all employees in a company," or "to plot the trajectory of a certain missile," or even "to pick compatible boy-girl pairs for the Senior Prom." Whatever the problem is, it can be seen as a specific application of the computer. Consequently, this kind of programming is usually called "applications programming."

There is another kind of programming that is quite different from applications programming, and quite essential to the effective functioning of the computer system. It is called *systems programming.*

If you have written any applications programs, you are probably aware that the computer is equipped with software, or programs, that make the operation of the computer more automatic, that make programming easier, and that increase the speed and efficiency of the hardware. These programs are the ones that are developed by systems programmers. The sole function of a systems program is to make the computer system operate more smoothly, or to make it easier to program. A systems programmer writes programs for the computer itself, while an applications programmer writes programs to solve problems for computer users.

There is another way to see the difference between the two kinds of programming. An applications program is written specifically to input certain information, process it, and output new information. Once the program has been executed, it may be removed from main storage and filed somewhere until it is needed again.

In contrast, systems programs do not produce output reports. They do not process information to produce new information. Instead, they act "behind the scenes" scheduling and allocating various pieces of hardware, keeping track of what is going on, or, in the case of a large computer, even permitting it to execute several applications programs simultaneously. Systems programs are usually in main storage all the time or are quickly accessible.

In summary, you can see that while applications programmers work with programs to process and produce useful information, systems programmers work with programs which make it easier for the computer to deal with those applications programs. These systems programs constitute the software of the computer system.

## SYSTEMS SOFTWARE

### Before software

As you know, in the early days of computing, computers were used in a very primitive way compared with present day standards. All programming was done in machine language. That is, each single instruction to "add" or "store" or "print a digit" had to be translated by the programmer to the numeric code that computers understood. Thus, writing programs was a slow and laborious task.

To give you an idea of how much more work there was for the programmer, look at how a simple addition program can be written in BASIC:

```
10 LET S = A + B
20 PRINT S
30 END
```

You can see how close this language is to the language we ordinarily think and speak in. This makes it easy to remember and to use. Now, compare this with the machine language that programmers used to have to use. The above addition program, for example, might look like this in machine language:

```
034
035
134
235
636
536
900
```

In the days before software, writing programs was called "coding." Programmers often were called "coders." You can see why!

Running programs on the computer was no less slow and laborious than coding the programs. The operator would use a "loader" program (already stored in memory) to read in the machine language instructions from cards or paper tape. Once the program was loaded, the operator would start the run. He or she might have to interrupt the run several times to prepare input or output media, load data, monitor the operation of the machine, or even make manual changes in the program. Mostly, programmers and operators worked very hard, and the machine waited.

## The first software: translators

The first real advance in using computer hardware came when a weary programmer realized that the computer itself, if properly programmed, had the ability to translate instructions (like "add" or "store" or "print a digit") to numeric code. Thus, the first "software" was developed: an automatic translator program. This meant that programmers could write programs in languages much closer to their own languages and have the computer translate them. These new programming languages are called "high-level languages" (or "procedure-oriented" languages) as opposed to "lower" level languages such as machine and assembly languages.

To this day, the automatic translation of human language program instructions into machine language code is one of the most important functions of software. Of course, we have become very much more sophisticated in developing translators. Today programmers can write instructions almost as they would talk, without worrying about the particular numeric code the machine uses. Modern day programmers can, therefore, spend much more of their time in analyzing problems and devising a solution, and much less time actually writing the instructions of the program than their counterparts of a decade or more ago.

## The un-automatic computer

After it was recognized that computers could do their own translating or coding of programs, the next obvious inefficiency in using a computer was tackled. That was the problem of how much time it took to execute a series of programs or "jobs." The steps shown next used to be involved when the computer was used to translate and then to run a program. You can see that there was not much "automatic" about the process. For a typical program to be run, the sequence used to be roughly like this:

- The programmer submits a program to be run, punched on cards, and some data to be processed, also on cards. (The program must be translated to machine language by the computer before it can be run.)
- The operator finds the stack of cards for the translator program, places these cards in the card reader, and loads the translator into the machine. The operator then puts the cards away.
- The operator takes the original program itself and places the cards in the card reader. Then the operator starts ("runs") the the translator program.
- The translator converts the original program to machine language and produces a stack of cards punched with the coded machine language version of the original program.
- The operator takes these machine language cards, puts them in the card reader, and reads them into the computer along with the data cards.
- The operator runs the program.
- When the program is finished (if there was no problem requiring operator intervention), the operator takes the entire set of cards from the reader, takes any output such as paper from the printer, puts these all together, and returns them to the programmer.
- The operator then starts the whole process again for the next program.

You can see that, even with automatic translator programs, operating a computer was still awkward, inefficient, and slow. It required many manual operations. This was true because the computer's main storage is usually too small to hold the original high-level program, the translator, and the machine language version of the original program all at the same time: The next step toward truly automatic computing had to be the invention of new hardware: auxiliary storage devices like magnetic tape and disk.

Once auxiliary storage devices were available, it was possible to simultaneously store everything required for translation. The translator could be kept in auxiliary storage. As the original program was read into main storage, the necessary parts of the translator would be called into main storage to do the translation. The machine language version could be put in auxiliary storage as it was produced by the translator.

### Automatic operating systems

The invention of this new hardware made it necessary that new software be developed, to control the interaction of the computer with auxiliary storage devices. This new software was called an *operating system*. The operating system automatically controlled all of the operations involved in translating and running a series of programs: reading in a program, calling on the required translator one piece at a time as needed, storing the translated machine language version, reading in the next program in the series, and so on.

Then, with new hardware and software (auxiliary storage and operating systems), the truly "automatic" computer became a reality. This meant that the steps involved in translating and running a program were reduced to the following:

- Programmer brings in a stack of cards containing a program and data.
- Operator places "job control cards" in front of the stack. These control cards contain information for the operating system, including:
  - the name of the program and perhaps of the programmer
  - which of the available translators is needed for this particular program
  - the type and format of the data
  - the special subroutines * and utility programs * the program calls for
  - which input and output devices will be used
- The operator places this stack of control cards, program, and data (called a "job") behind the larger stack which is already in the card reader. There are now many jobs lined up to run.
- The operator starts the computer.

* These are standard routines, such as finding a square root, or alphabetizing a list. They are stored and called on whenever they are needed in a program.

While all of the jobs are being run, one after the other, the operating system types messages, telling the operator which input and output devices to have ready. The operator can then be mounting tapes, changing printer paper, or loading more cards without stopping the operation of the computer.

Before going on, check your understanding by answering the check questions.

## Check your understanding

**1.** In the early days of computing, the programmer was called a "coder" primarily because

a. the programmer wrote programs in abbreviated words like STO and ADD.
b. the programmer was required to understand Morse code.
c. the programmer used translators to develop machine codes.
d. the programmer programmed in the numeric code of the machine.

**2.** In the early days of computing, the first "software" developed was

a. an automatic translator program.
b. an operating system.
c. a job control language.
d. a machine language.

**3.** The lowest-level programming language, and the most difficult to use for writing programs, is

a. assembly language.
b. translator language.
c. machine language.
d. BASIC.

**4.** What two hardware and software developments made operating the computer truly automatic?

a. Translators and main storage
b. Auxiliary storage and operating systems
c. Auxiliary storage and translators
d. Card readers and operating systems

## THE FUNCTION OF SOFTWARE,

You have now seen the two most important functions of software:

- To provide aids for the programmer, to make the job easier and quicker.
- To make the operation of the computer more efficient, speedy, economical and automatic.

The two functions are accomplished by two different kinds of software: (1) translator programs (to aid programmers) and (2) an operating system (to increase the operating efficiency of the computer). The systems programmer works in the main with these two kinds of software—either creating new translator and operating systems or installing and modifying those already available. Let's take a closer look, then, at each of these two types of software. First, we'll consider translators.

### Translators

A translator, as you already know, has one purpose. It is to convert instructions written in a language understandable by humans to the machine language understandable by the computer.

Every computer has its own unique machine language, like no other computer's machine language. This means that every brand and model of computer will have translator programs that convert high-level language programs into its particular machine code and no other.

When stored inside the computer, the symbols which make up the machine language program are 1's and 0's. For example, a typical program instruction to "print out a number" might be

```
010100110110
```

A program to add two numbers which are in storage and print the result might look like this:

```
000000100010
000000100011
000010000110
000011101011
001001111100
001000011000
001110000100
```

This is obviously not an easy language to learn or work with!

Without much effort, the computer can be wired so that, instead of typing in or out in long strings of 1's and 0's, we can use a machine language of more normal-looking numbers. Even so, the preceding program is difficult to read:

```
034
035
134
235
636
536
900
```

Although it is easy for the computer to work with machine language, it is difficult for people to program in this language.

*Assemblers.* However, the computer has the power to recognize letters and symbols, as well as numbers. It can be programmed to substitute the number 2 every time it receives the symbol ADD, and substitute the number 6 every time it receives the symbol STO. In this way, the machine can be programmed to translate a program written alphabetically into its equivalent machine code, automatically. This type of translator is called an *assembler.*

Let's look again at the program to add two numbers and print the result. Examine the program as a programmer would think it, and then as a programmer would write it in assembly language. Then look at the machine language program as translated by the assembler.

| *The desired instruction is as follows:* | *The programmer writes in assembly language:* | *The assembler translates to the following:* |
|---|---|---|
| Input "A" | INP A | 034 |
| Input "B" | INP B | 035 |
| Clear accumulator register and add "A" | CLA A | 134 |
| Add "B" ("S" is now in accumulator) | ADD B | 235 |
| Store "S" | STO S | 636 |
| Output (Print) "S" | OUT S | 536 |
| Halt and reset | HRS | 900 |

The translation from assembly language to machine language is a fairly simple conversion. When using assembly language, however, the programmer still must know how the machine works in order to tell the machine exactly what to do at every step. Each instruction the programmer writes in assembly language is translated into exactly one machine language instruction in a one-to-one conversion.

Assembly language is the programming language most frequently used by systems programmers. It allows the systems programmer to deal directly with the machine in very fine detail without the nuisance of translating to machine code.

In fact, systems programmers may deal with assembly language in two ways. They will *use* assembly language to write most of their programs, but they may first of all *create* the language and the assembler program which will translate the assembly language into the machine language of the computer. Of course, when they are creating the assembler, they must program in machine language since no translator yet exists. Machine language and assembly languages are very closely related, as you can see. While the machine code is created by the engineers and the assembly language for the computer is created by the systems programmers, both are unique to the particular model of computer. Neither the machine language nor the assembly language used in one model of computer can be used to program another model.

*Macro-instructions.* Often a programmer finds a certain set of machine language instructions is used over and over. For example, a particular computer might need a particular set of three instructions, one after the other, to print a number. In a program with lots of printout, these three instructions might be used hundreds of times, always in the same order.

Rather than force the programmer to write these same three instructions over and over every time he or she wants to print a number, a short sub-program might be included inside the assembler to take care of these instructions. Every time the programmer gives the instruction PRINT in assembly language, for example, the assembler would translate it into the necessary set of three instructions.

Any instruction, such as PRINT, which gets translated into several machine language instructions, is called a *macro*-instruction. There might be many such macro-instructions permitted by the assembler.

Although the use of macro-instructions greatly simplifies programming, reduces the length of the original program, and cuts down on errors~~, it adds much work for the computer. This is true because the~~ translation here involves more than just changing each single word into a single number.

The assemblers created and used by systems programmers usually incorporate many macro-instructions. However, they are still called assemblers, because the resulting machine language version of the program still has more or less one machine language instruction for each ~~instruction in the original program.~~

*Compilers.* Unlike assembly language, the high-level languages such as BASIC and FORTRAN used by programmers are made up of nothing but macro-instructions. Consequently, for every instruction the programmer writes in such a language, there needs to be a translator which will produce not one but many machine language instructions. A translator program that makes this one-to-many conversion is called a *compiler* program.

You have probably used compilers in your applications programming. There are many available. Some are BASIC (Beginner's All-Purpose Symbolic Instruction Code), FORTRAN (FORmula TRANslation), COBOL (COmmon Business Oriented Language), and ALGOL (ALGOrithmic Language). As the names imply, each of these languages was designed to be useful in solving a particular type of problem, and may even be useless in other types of problems. BASIC, for instance, was intended to be a simple language for beginning programmers to learn and use. Consequently, it is not as sophisticated in formatting input and output for reports as, say, COBOL. While COBOL was designed for business applications, it is not the best language to use for mathematical or scientific calculations. FORTRAN and ALGOL are better languages to use for applications of that type.

These languages and others like them are often called *procedure-oriented* languages. Each one is oriented for use in a particular field—beginning programming, business, science, etc.

It is important to understand that these high-level languages exist independently of any particular machine. That is, the words, symbols and rules used in the FORTRAN language are essentially the same no matter what computer you are using. The FORTRAN *compiler*, however—the actual translator program—will be different for each machine, because each machine has a different machine language.

Similarly, an "assembly *language*" is a standard set of words and symbols, with rules for putting them together. An "assembler" is the translator program that converts those words and symbols into numeric machine language.

Now check your understanding.

## Check your understanding

1. Two kinds of software which systems programmers develop are translators and operating systems. Translators aid the programmer, while operating systems

   a. are used to make programming easier.
   b. make the operation of the computer most efficient.
   c. are written guidelines for operating the computer.
   d. are the systems used by the engineers to build the computer.

2. The programming language most frequently used by systems programmers is

   a. machine language.
   b. assembly language.
   c. high-level languages like BASIC and FORTRAN.
   d. macro-instruction languages.

3. An assembler is a software program which translates

   a. from a high-level language like FORTRAN into machine language.
   b. from machine language into assembly language.
   c. from assembly language into a high-level language.
   d. from assembly language into machine language.

4. Which of the following is true of both machine language and assembly language?

   a. They differ from computer to computer.
   b. They are the same from computer to computer.
   c. They are standard problem-oriented languages.
   d. They are created by systems programmers.

5. A macro-instruction is

   a. one instruction which must be translated into two or more machine language instructions.

b. several instructions together which must be translated into one machine-language instruction.
c. one instruction in any language which is used very often.
d. one instruction in any language which can be used to mean many different things.

6. A compiler is a translator program which

a. converts assembly language into machine language.
b. converts problem-oriented languages into machine language.
c. translates machine language into BASIC.
d. translates assembly languages into high-level languages.

7. Which of the following is *not* true about problem-oriented languages?

a. They include BASIC and COBOL.
b. They are made up of macro-instructions.
c. They are translated by compilers.
d. They are designed for use on one particular model of computer.

## The operating system

An operating system is a collection of programs designed to increase the productivity of the computer. Its individual programs automate much of the preparing, loading, and execution of applications programs on the computer equipment.

Because the operating system controls and coordinates so much of the computer system's operation, it usually frees the computer operator from any need to intervene during a run. This means that when an operating system is available, the computer operator can usually use the time during one run to set up the equipment for the next run. In this way, several jobs can often be run in an uninterrupted sequence, under the control of the operating system.

The kinds of software programs used in operating systems vary from system to system, but the ones most often found are

- A supervisor program (often called the "monitor" or "executive")
- Input/output control programs (or routines)
- Utility programs (or routines)

These are fundamental programs in an operating system, although different systems may call them by different names. Since a complete operating system may include several programs in addition to these, when you have a chance to work on a real computer system, you should read the manuals that accompany that system carefully. The manuals should give you complete descriptions of the software in the system.

As is true for all computer software, the programs in the operating system are developed by the manufacturer's systems programmers. But, the user's systems programmer needs to be thoroughly familiar with the programs. Then he or she can use (and train the user's applications programmer to use) and adapt the system to work more efficiently for the user's particular purposes.

Now let's look at the fundamental kinds of programs in the operating system to see just what each one does to increase the efficiency and ease of operation for both the computer and the computer operator.

***The supervisor.*** The supervisor program or routine is the most important and complex of all programs in the operating system. It sets up, monitors, and controls the operation of other programs. While most software is placed in auxiliary storage (disk or drum) until needed, the supervisor routine usually is loaded into the computer each morning and remains in main storage throughout the day's operation. It organizes and supervises the running of all other programs in the operating system.

A basic supervisor routine controls the execution of a program by clearii he storage area (placing zeros or blanks in work areas), then loading the program into storage.

Once the program is running, other programs in the operating system take care of the flow of operation. When errors are encountered during processing—for example, when a program exceeds the available storage capacity—control is once again transferred to the supervisor routine which takes corrective action or halts execution.

An important part of the supervisor program is the "job control" routine. This routine controls the execution of a series of programs sequentially with a minimum of operator intervention. To do this, the operating system requires initial instructions. The programmer usually has these instructions punched on cards and gives them to the computer operator for the run. These cards are called the *job control cards*. The instructions on the job control cards indicate what programs and subroutines in the operating system are to be used, which translator is required, the type and format of data, and which input/output devices will be used.

The supervisor routine starts and stops the execution of each program and performs many functions otherwise done by the operator, but it does them faster and more accurately. Where operator intervention is required, the supervisor will print out operator instructions on the console typewriter.

In some more complex, modern computer systems, the central processor is a *multiprocessor*. That is, it has more than one arithmetic-logical unit. This means that several programs can be run simultaneously. Of course, the supervisor routine and its job control programs must be even more sophisticated to schedule and monitor the execution of more than one program at once.

In other systems, a technique called *multiprogramming* is used where there is only one arithmetic-logical unit, but several programs are in the computer at one time. The supervisor routine divides the computing time among the different programs, transferring control from one to another to process them all at the same time.[*] Again, a sophisticated supervisor routine is required to schedule and control the simultaneous execution of the programs.

As an example of multiprogramming, a program to read and print cards may be run at the same time as a program to sort records on magnetic tapes. The supervisor routine initiates the instruction to "read a card" for the first program, then may transfer control to the sort program. When a signal indicates that the card has been read, the second program would be interrupted and control returned to the read-print program for the next step—that of moving the data into position for printing. Since the read-print program can proceed no faster than the reader and printer can operate, the central processing unit (CPU) would be idle much of the time if the supervisor routine did not use this idle time to run another program.

*Input/output control programs.* The control of input and output (I/O) operations requires a separate set of programs because these operations are mechanical functions performed by mechanical devices. If the speedy electronic central processor had to stop and wait during each I/O operation, its efficiency would be greatly reduced.

[*] Notice that multiprogramming refers to the way the computer works with programs rather than to a programming technique.

If data were read directly into the CPU, or directly out, the fast CPU would be tied up as the slow mechanical reader or printer operated. However, most computers now use electronic *buffers* to cut down this delay. When data is read from an input device, it goes into an input buffer, where it is held until the CPU is ready for it. Then it is transferred from the buffer to the CPU to the output buffer. It is output from the buffer when the output device is ready.

The scheduling and overlapping of these buffered I/O operations are complicated and must be controlled by the input/output control programs, part of the operating system.

These control programs also handle such operations as detecting I/O errors, switching to an alternate tape drive at the end of a reel, and searching for the proper items to be input from a reel of tape, a disk, or a drum.

*Utility programs.* Utility programs aid in the production work of a computer center by servicing jobs which recur frequently. While the specific utility programs available will differ from one system to another, a fairly complete set would include the following:

- A program loader
- Programs to transfer data from one medium to another (card-to-tape, tape-to-card, card-to-disk, disk-to-card, tape-to-printer, disk-to-printer, etc.)
- "Diagnostics" or test routines to help identify and correct machine failures or program errors
- Special programs to protect permanent files, sort and merge data, duplicate tapes or disks onto new tapes or disks, and perform "memory dumps" (output the entire contents of main storage on the printer)

The *program loader* is used to enter new programs into storage. Even the programs in the operating system must be initially introduced into storage by the loader.

A program is stored in memory in the same way as data. It is read in, probably from punched cards or tape. The program loader supervises the reading in and allocates storage locations for each program instruction read in. Finally it transfers control to the first instruction of the program just loaded, so the program can be executed.

You may be asking, "But how does the loader *itself* get into storage?" It's a good question.

A brand new computer, with a completely blank memory, must have the loader stored by hand. This is a tedious process, since the operator must store each single instruction of the loader by hand, accessing main storage directly through the switches on the console control panel.

To save time, a shorter, more primitive loader called the "bootstrap loader" is usually stored by hand. Then the bootstrap loader is used to automatically read in (or "load") the official program loader, which is considerably longer. Once the program loader is stored, it is used to load all the programs.

*Transferring data* from one medium to another is often necessary. It must be controlled by special programs such as a card-to-tape program, tape-to-printer program and the like. Likewise, there are special programs for duplicating tapes or disks onto new tapes or disks.

*Diagnostics* or *test routines* may function automatically to check for machine or program error during operation of programs. They also may be called into action when there appears to be a malfunction in a program. If a programmer wishes to use a test routine designed to help locate a program error, she or he would probably call for a "trace" or a "dump."

A trace routine will print the results as each step of the program is executed. This allows the programmer to closely observe the operation of the program.

A dump routine, often called a "memory dump" or "memory print," will print the contents of each storage location at any specified check point in a program. Programmers can also specify the boundaries of the storage area they wish to examine.

*Special programs* take care of frequently performed operations, such as copying all files from tape or disk at intervals as a protection against accidental erasure or loss of important data.

Another frequently used utility program found in most systems is a sort-merge routine. It should be able to sort either numeric or alphanumeric records into ascending or descending order, merge one set of records with another set of records, add and delete data, and provide for input and output on various media.

# Check your understanding

1. An operating system

   a. allows several jobs to be run in an uninterrupted sequence.
   b. saves the computer operator from much manual intervention during computer operation.
   c. is one very large program.
   d. includes as one of its programs a "supervisor."
   e. is a collection of programs.

2. A supervisor

   a. monitors and controls the operation of all other programs in the operating system.
   b. takes over when errors are encountered in processing.
   c. starts and stops the execution of each program.
   d. is often called a monitor or executive.
   e. all of the above.

3. What is the main function of the input-output control programs?

   a. To mount and dismount tapes, disks, and drums, to sort cards, and so forth
   b. To replace the electronic input and output buffers
   c. To control the execution of two or more programs, transferring control from one to another
   d. To schedule input-output operations, detect I/O errors, and search out input items from auxiliary storage media

4. Which of the following could be included in a typical set of utility programs?

   a. A FORTRAN compiler
   b. An I/O control program
   c. Programs to sort and merge data, duplicate tapes or disks, etc.
   d. A program loader
   e. Applications programs
   f. Test routines
   g. Programs to transfer data from one medium to another
   h. An executive program

**5.** Which of the following best describes the program loader?

a. An input-output device designed especially for feeding programs into the computer
b. The switches on the computer's control panel which are used to load programs directly into main storage
c. A program which supervises the reading in and storage of programs and which transfers control to the program's first instruction
d. A computer operator who specializes in loading programs into computers by hand

## SYSTEMS PROGRAMMING

### Introduction

The techniques and tools used in the work of the systems programmer are the ones common to all programming work. They include the use of various special programming languages, diagrams and flowcharts, and debugging and documentation techniques. As you will see from the discussion below, however, the way systems programmers use these techniques differs somewhat from the way they are used by applications programmers.

### Programming languages

One of the programming languages used most often by systems programmers is machine language, which, as you know, is particular for each model of computer. Once on the job, systems programmers naturally become expert specifically in those machine languages used by the particular computers they deal with.

You will remember that, of all programming languages, machine language is the most tedious and time consuming to use. The programmer must translate every instruction to the numeric code of that particular computer before loading the program. Consequently, assemblers for computers are usually the first part of software developed. From then on, both the manufacturer's and the user's systems programmers can use assembly language.

Assembly language is also individual to each computer model. While the machine language is devised by the engineers of the computer, however, the manufacturer's systems programmer may create the special code of the assembly language for the computer he or she is developing software for.

You may be wondering, "Why don't they use BASIC or FORTRAN?" The reason is, system software must be as efficient as possible in the amount of storage space it requires and must run in the fastest possible time. Writing a program in a high-level language like FORTRAN, which must be compiled, reduces the efficiency of the translated machine language version.

Systems programmers are so familiar with the characteristics of each machine operation, the timing of each operation, and the most efficient combination of operations, that they can make the software much more compact and speedy by writing it in machine or assembly language.

Systems programmers must nevertheless be thoroughly familiar with the commonly used high-level languages (BASIC, FORTRAN, COBOL, etc.). This is so they will be able to design the high-level language compilers most frequently required in the software package.

## Flowcharting techniques

You will recall some of the typical flowcharting symbols used by programmers, as shown in Fig. 5-27.

Since systems programmers usually deal with very complex programs, they draw a macro-flowchart first, showing the main steps and parts of the program. Then they may draw more detailed flowcharts,

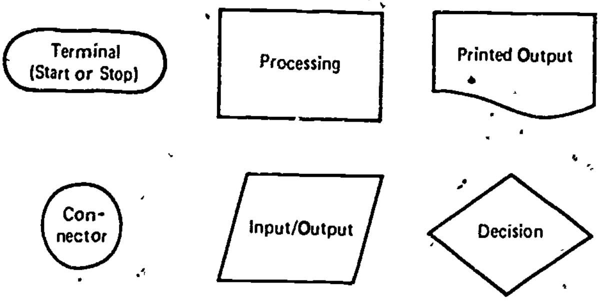


Fig. 5–27 Flowcharting symbols

progressively breaking down the problem into smaller and smaller steps. Finally, they will draw a micro-flowchart that the systems programmer uses as a guide in writing the program in machine or assembly language.

To see the difference between a macro-flowchart and a more detailed flowchart, examine the example shown in Fig. 5-28. In this figure, a macro-flowchart is shown for a program that computes grade point averages. Then in Fig. 5-29, one step in the macro-flowchart is broken down into a detail flowchart.

A still finer level of detail could be achieved by taking one step from the detailed flowchart in Fig. 5-29, and breaking it down into a micro-flowchart, showing every single step required for the computer to add all of the stored products.

ADD ALL
OF STORED
PRODUCTS

Although systems programmers would probably not develop programs to compute grade point averages, they would use the same techniques illustrated in Fig. 5-28 and Fig. 5-29 to draw macro- and micro-flowcharts.

## Debugging the program

You will recall that a programmer, once the program has been completed, must test and debug that program before it can be used.

When an applications programmer is testing and debugging, this procedure is often followed:

1. Enter the program—which is then compiled.
2. Receive a list of "error messages."
3. Run a special debugging program which gives a printout (or "dump") of the contents of storage and traces the execution of the program to help locate "bugs."
4. Take the error messages, storage dump, and trace to his or her desk to search for the bugs.

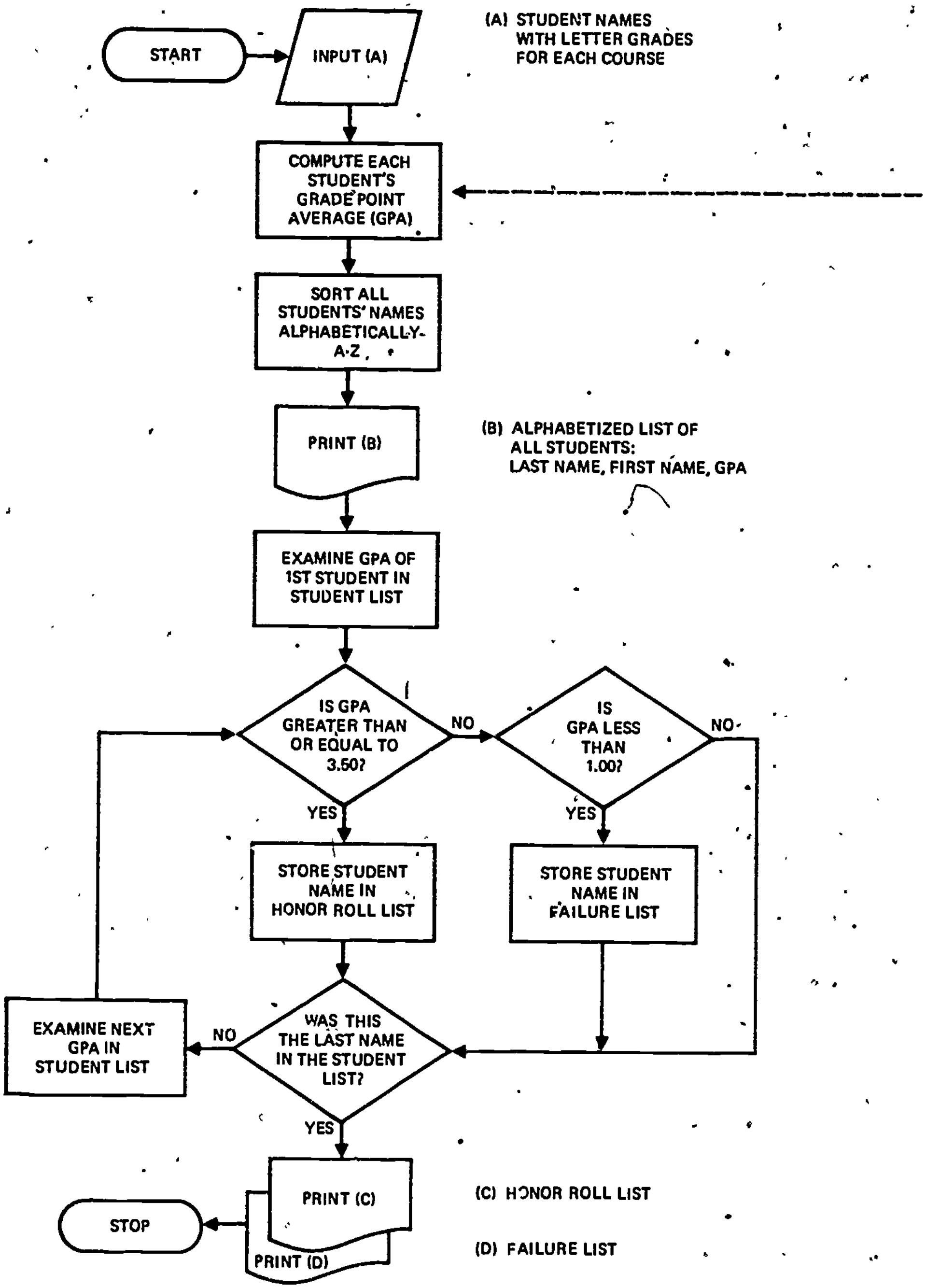

START
INPUT (A)
(A) STUDENT NAMES WITH LETTER GRADES FOR EACH COURSE
COMPUTE EACH STUDENT'S GRADE POINT AVERAGE (GPA)
SORT ALL STUDENTS' NAMES ALPHABETICALLY-A-Z
PRINT (B)
(B) ALPHABETIZED LIST OF ALL STUDENTS: LAST NAME, FIRST NAME, GPA
EXAMINE GPA OF 1ST STUDENT IN STUDENT LIST
IS GPA GREATER THAN OR EQUAL TO 3.50?
NO
IS GPA LESS THAN 1.00?
NO
YES
STORE STUDENT NAME IN HONOR ROLL LIST
YES
STORE STUDENT NAME IN FAILURE LIST
EXAMINE NEXT GPA IN STUDENT LIST
NO
WAS THIS THE LAST NAME IN THE STUDENT LIST?
YES
PRINT (C)
PRINT (D)
STOP
(C) HONOR ROLL LIST
(D) FAILURE LIST

Fig. 5–28 Macro-flowchart of G. P. A. program

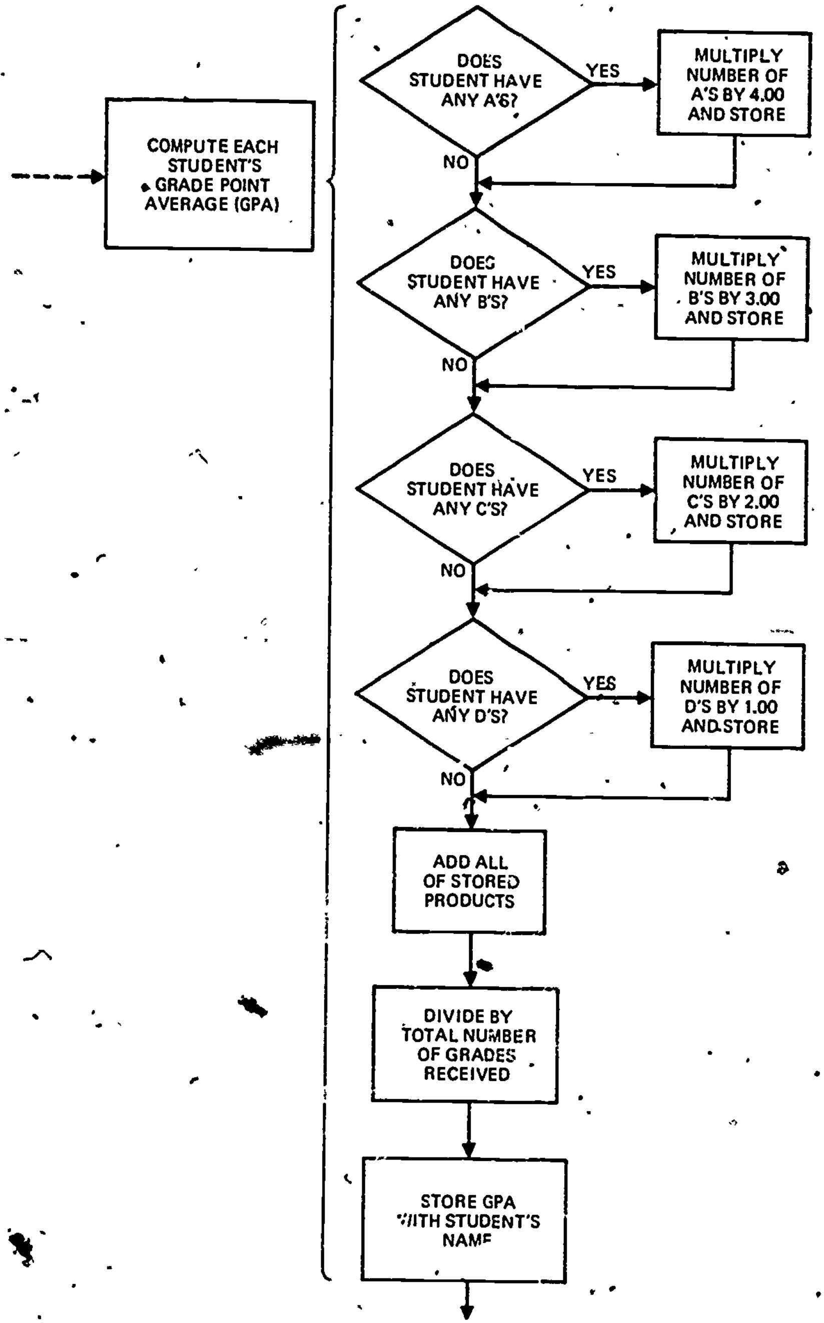

COMPUTE EACH STUDENT'S GRADE POINT AVERAGE (GPA)
DOES STUDENT HAVE ANY A'S?
YES
MULTIPLY NUMBER OF A'S BY 4.00 AND STORE
NO
DOES STUDENT HAVE ANY B'S?
YES
MULTIPLY NUMBER OF B'S BY 3.00 AND STORE
NO
DOES STUDENT HAVE ANY C'S?
YES
MULTIPLY NUMBER OF C'S BY 2.00 AND STORE
NO
DOES STUDENT HAVE ANY D'S?
YES
MULTIPLY NUMBER OF D'S BY 1.00 AND STORE
NO
ADD ALL OF STORED PRODUCTS
DIVIDE BY TOTAL NUMBER OF GRADES RECEIVED
STORE GPA WITH STUDENT'S NAME

**Fig. 5–29** Detaile flowchart of one step from the G. P. A. program macro-flowchart

5. Examine both the macro- and micro-flowcharts for errors in logic, and use the error messages, dump, and trace to help track down errors.
6. Correct all the errors found and then re-enter the program and compile it.
7. Continue this procedure until the program compiles and runs correctly.

A systems programmer debugs programs using a somewhat different debugging technique. For one thing, the automatic debugging aids (error messages, storage dumps, traces, etc.) are usually not available during the development of basic software. For another, the systems programmer is probably programming in a low-level language.

Consequently, systems programmers use a debugging procedure, following this general outline:

1. Load the machine or assembly language program. (In the case of assembly language programs, use the assembler to translate it into machine language.)
2. Execute the program. If it does not run correctly, proceed to steps 3 and 4 below.
3. Execute the program one step at a time, using the control panel to follow the program execution and to pinpoint errors as they occur.
4. Correct errors, repeating step 3 until the complete program runs correctly.

As you can see, for the applications programmer, debugging is more of a desk exercise and bugs are more easily found. The systems programmer is more likely to do the debugging directly on the computer. It is likely to take more time to detect and to correct errors in programs of the systems programmer.

## Check your understanding

1. Assemblers are usually the first software developed for a computer because .

   a. machine language is tedious, time-consuming, and complicated.

b. assembly language is easier to use than machine language.
c. assembly language is more commonly used than a high-level language for software development.
d. all of the above.

2. High-level languages like FORTRAN, COBOL, BASIC, etc., are seldom used for development of system software because

a. they are difficult to use.
b. system software must be very efficient in using storage space, and running time.
c. systems programmers do not know how to program in high-level languages.
d. the timing of each operation is off.

3. Which of the following flowcharts is the easiest one to read?

a. Macro-flowchart
b. Micro-flowchart
c. Systems flowchart
d. None of the above

4. Which of the following flowcharts would be the easiest to use as a guide in writing an assembly language program?

a. Macro-flowchart
b. Micro-flowchart
c. Systems flowchart
d. None of the above

5. For which of the following kind of programmer is debugging his or her programs usually more difficult?

a. Applications programmer
b. Systems programmer
c. FORTRAN programmer

6. Since automatic debugging aids are not usually available during the debugging of basic software, the systems programmer usually uses which of the following to pinpoint errors during the debugging process?

a. The computer control panel
b. Error messages listed by the computer
c. The "dump" printout
d. Both b and c above

## DOCUMENTATION OF SOFTWARE COMPONENTS

The manufacturers' systems programmers are usually responsible for providing two kinds of documentation for the software components they develop. One kind is a complete set of flowcharts, diagrams, and specification tables, accompanied by a complete listing of the programs. These constitute the basic documentation of the software.

The second kind is a complete textual description of the software components including illustrative flowcharts and diagrams and full explanations of how each element of the component works and how it can be used.

### Basic documentation

The basic documentation of a system is usually quite standard, including

- all logic diagrams and macro- and micro-flowcharts which detail the component
- all specification tables giving the needed code translations, storage locations, and language and programming keys
- a complete computer listing of all the programs in the component.

The diagrams, flowcharts, and tables are ordinarily polished versions of the working charts and tables the systems programmer ~~used~~ while developing the component. Since the program listings used in the basic documentation are printed out by the computer itself, you can see that basic documentation is not difficult for the systems programmer to prepare.

### Reference manual

The second kind of documentation which provides descriptive accounts of the system generally takes the form of a complete reference manual for the component. It is this documentation which most often serves as the reference material for users of the software.

The techniques used by systems programmers to write their reference manuals differ widely. Since the purpose of the manuals is always to show or explain clearly what the software does and how to use it, there are several features which are characteristic of most reference manuals.

- A preface or introduction to the component is usually provided, along with a table of contents to make the manual easy to use.
- The first part of the manual usually presents a description of the software components as a whole. It explains how the particular computer system operates with the components.
- The central portion of the manual usually presents the components of the software in logical order. Each program is introduced with a clear description of its purposes, capabilities, and singularities. Specifications for using the programs are nearly always provided.
- A complete reference manual will usually include a separate section on the special programming techniques either necessary or useful when programming with the system.
- Finally, a section is normally devoted to a description and discussion of the various subroutines and programs in the system's library.
- As a last aid to the user, appendices are often added giving any tables or listings which would be especially useful to the user, such as tables showing all error messages the system is designed to produce and what they mean and/or listings of programs.

Obviously, the production of this kind of documentation requires quite a bit of special work on the part of the manufacturers' systems programmers. However, since they have developed the software being documented, and know it thoroughly, the writing of this reference material usually involves just logical organization of the information and clear writing. The more effective the organization and writing are, of course, the more useful the manual will be as a user's reference guide.

Clearly, the documentation techniques described above are used primarily by the manufacturer's systems programmers who develop the software packages and the documentation to go along with it. The user's systems programmers generally only need to understand these documentation materials so that they can effectively use the software and train the user's application programmers to use them.

It often happens, however, that the user's systems programmer has occasion to adapt or change some part of the software to fit the specific needs of her or his own company. In all such cases, of course, the user's systems programmer will naturally produce basic documentation for the changes. A description and guide to the adapted program or programs for use by the applications programmers in the company may also be written.

## Check your understanding

1. Basic documentation is which of the following?

   a. A user's reference manual
   b. An engineering report
   c. The user's first printouts from the system
   d. None of the above

2. The production of basic documentation differs from the production of reference manuals in which one of the following ways?

   a. It involves much more writing of descriptions and organization of materials.
   b. It is usually easier and more automatic since most of the materials have already been produced in some form during the programming process.
   c. Since it is used only by the systems programmer who developed the software, it can be in rough and incomplete form.
   d. None of the above are true.

3. The reference manuals produced by systems programmers usually contain

   a. only flowcharts, program listings, and reference tables for programmers.
   b. complete descriptions and guide materials to each component in the software system.
   c. dump printouts.
   d. none of the above.

4. Which of the following would be considered the most useful in learning how to use a new software system?

   a. Basic documentation
   b. Engineer specifications
   c. Computer runs
   d. Reference manuals

## THE SYSTEMS PROGRAMMER AT WORK

To gain a better understanding of the job, let's take a look at a systems programmer as he might function on a typical day. First, we visit him in the laboratory of the manufacturer.

### The manufacturer's systems programmer

Mark Peetz is a systems programmer. For three years he has worked for International Computers, Inc. Before that he was a user's systems programmer in the computer center at the state university, where he worked while he was attending college and for an additional two years after graduation.

About two years after Mark came to ICI, he was called in by his supervisor, Katherine Raymond. The Market Research Manager, Ken Rowlins, was with her. Together, they told Mark about a new line of ICI computers to be developed, which they said would be called the Galaxy series. ICI had done a market survey, and Galaxy was the result. There would be fou: models in the series, to be called Galaxy One (the smallest), Galaxy Two, Galaxy Three, and Galaxy Four (the largest).

ICI was the fourth largest manufacturer in the computer world, with 8% of the market. The number one manufacturer, with 65% of the market, had recently announced that they were producing a new line of computers called the 500 series (Model 500/10, 500/20, and 500/30).

ICI had decided to compete vigorously with the 500 series by designing the new Galaxy line of hardware and software.

ICI's recent market survey showed that users who presently had the number one company's computers were disappointed in the software provided. If the Galaxy series included software superior to the 500 series, it should be able to win customers away from the number one company.

Thus, Mark was told, a gigantic advertising campaign was being launched by ICI. The slogan was "ICI has *User-Oriented Software* on its new Galaxy series."

This "user-oriented" software was to be developed by a team of systems programmers—and Mark's supervisor was in charge of that team.

For the first six months, while the engineers completed the hardware design and built prototype models, the programmers did their own preliminary work. Since a particular type of user had been identified as the "target market," the programmers learned all they could about the software needs of these users.

As it developed, most of the users preferred to program in the COBOL language most of the time, but often needed to use FORTRAN. The Galaxy Four was to have the additional capability of time-sharing, and the BASIC language was needed for that.

All of the programmers studied the flowcharts and diagrams that detailed the electronic logic of the Galaxy series. They learned to write programs in machine language—the numeric code unique to the Galaxy computers.

Meanwhile, the senior systems programmers, including Mark, specified exactly what programs would be developed and assigned the programming tasks. It was decided that the Galaxy series computers would be delivered to customers equipped with the following:

- An assembler
- A COBOL compiler
- A FORTRAN compiler
- An operating system, including a supervisor, I/O control programs, and utility programs

In addition to the above, the Galaxy Four model would have

- A BASIC compiler
- A time-sharing supervisor

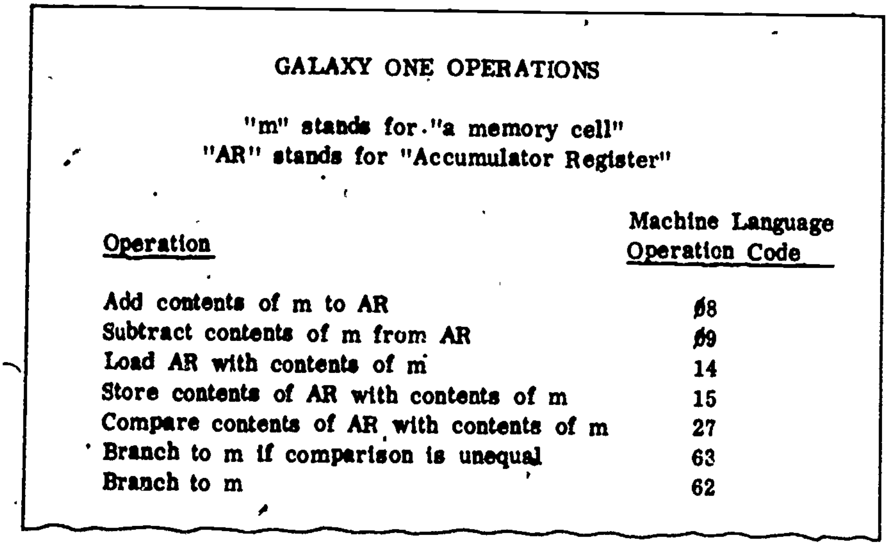
The first task was to write the assembler, so that all other software could be written in assembly language rather than machine language. The assembler would have to have several parts and several programmers would be assigned to create these parts. Mark was assigned to create the translator part of the assembler, the simplest part, himself.

***Developing an assembler.*** Let's look briefly at some of the steps Mark would have taken to create part of the translator. Then you will have some idea of what goes into just the simplest part of the assembler.

This translator part of the assembler is the one primarily responsible for converting all operations from assembly language into the particular machine language code the computer is wired for. In order to develop this translator for Galaxy One, then, Mark would first need to know exactly what operations the computer is designed to perform and what the machine language code for each of these operations is. A list of operations and operation codes would be provided by the engineers who designed Galaxy One. Here's an example of part of the list Mark might have received.

As you already know, assembly languages are made up of abbreviated words which are much easier than code numbers for programmers

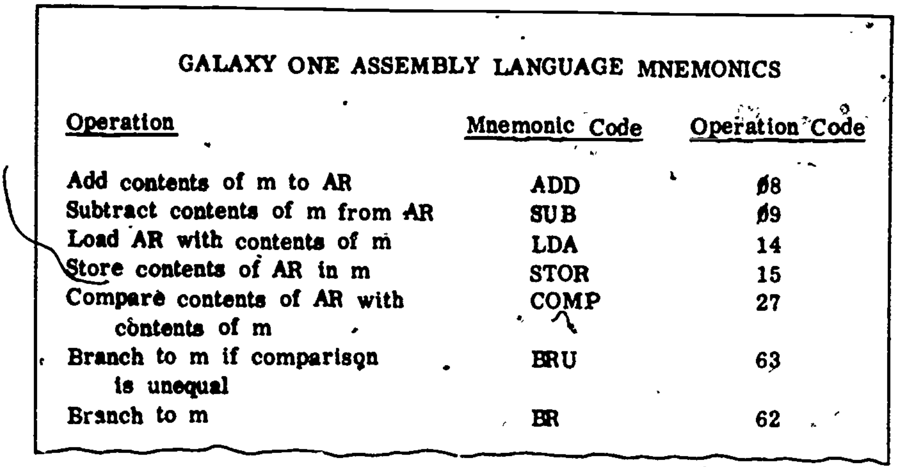
Fig. 5–30 Galaxy One operations



GALAXY ONE OPERATIONS

"m" stands for "a memory cell"
"AR" stands for "Accumulator Register"

| Operation | Machine Language Operation Code |
|---|---|
| Add contents of m to AR | ø8 |
| Subtract contents of m from AR | ø9 |
| Load AR with contents of m | 14 |
| Store contents of AR with contents of m | 15 |
| Compare contents of AR with contents of m | 27 |
| Branch to m if comparison is unequal | 63 |
| Branch to m | 62 |



GALAXY ONE ASSEMBLY LANGUAGE MNEMONICS

| Operation | Mnemonic Code | Operation Code |
|---|---|---|
| Add contents of m to AR | ADD | Ø8 |
| Subtract contents of m from AR | SUB | Ø9 |
| Load AR with contents of m | LDA | 14 |
| Store contents of AR in m | STOR | 15 |
| Compare contents of AR with contents of m | COMP | 27 |
| Branch to m if comparison is unequal | BRU | 63 |
| Branch to m | BR | 62 |

Fig. 5–31 Galaxy One assembly language mnemonics

to remember and use. The abbreviations used in assembly languages are usually called "mnemonic" codes. That is, they are designed to aid the memory. A mnemonic code normally has from two to six letters especially picked so they will remind the user of a whole series of words.

For instance, for the first operation, "Add contents of a memory cell to the Accumulator Register," a good mnemonic code would be "ADD." Fig. 5-31 shows the mnemonic codes which Mark might have invented for the Galaxy One operations shown in Fig. 5-30.

At this point, Mark would have a complete picture of all the operations Galaxy One could perform, the assembly language mnemonics to be used for these operations, and the machine language code corresponding to each mnemonic. So, he would be ready to develop the "translation flow" to be used in the assembler. To start with, he would probably draw up a flow diagram something like the one partly shown in Fig. 5-32.

With this flow diagram showing the large steps clearly, Mark would now begin to write the instructions necessary to do the translation of the operations.

The "first draft" of his translation program might start something like that shown in Fig. 5-33.

As an exercise, copy and complete on your copy the "flow of translation" diagram next to the translator program in Fig. 5-32. This should help you see clearly the "flow" that has been used in the program.

Once the translator has been completed, Mark would be almost ready to convert his program into machine language and to load it. But, first, he would have to assign a location in the computer to the mnemonic codes and operation codes used in the program.

For the first part of his program, he would need four storage locations for the words (ADD, SUB, LDA, and STOR) which any mnemonics being translated must be compared with. In addition, he would need four locations for storing the operation codes (08, 09, 14,

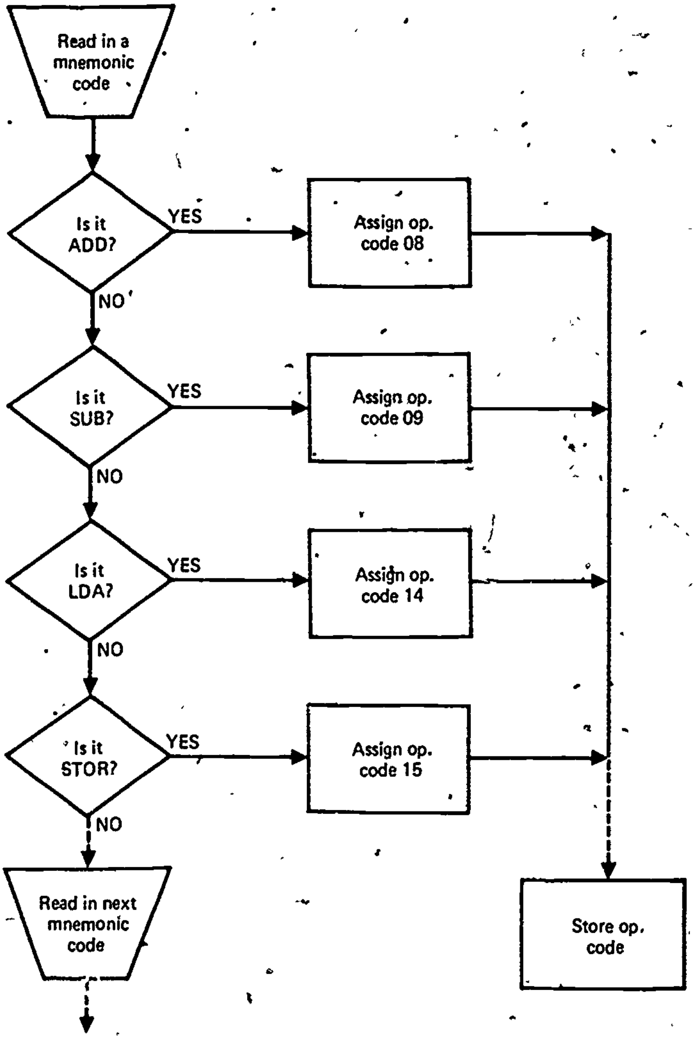
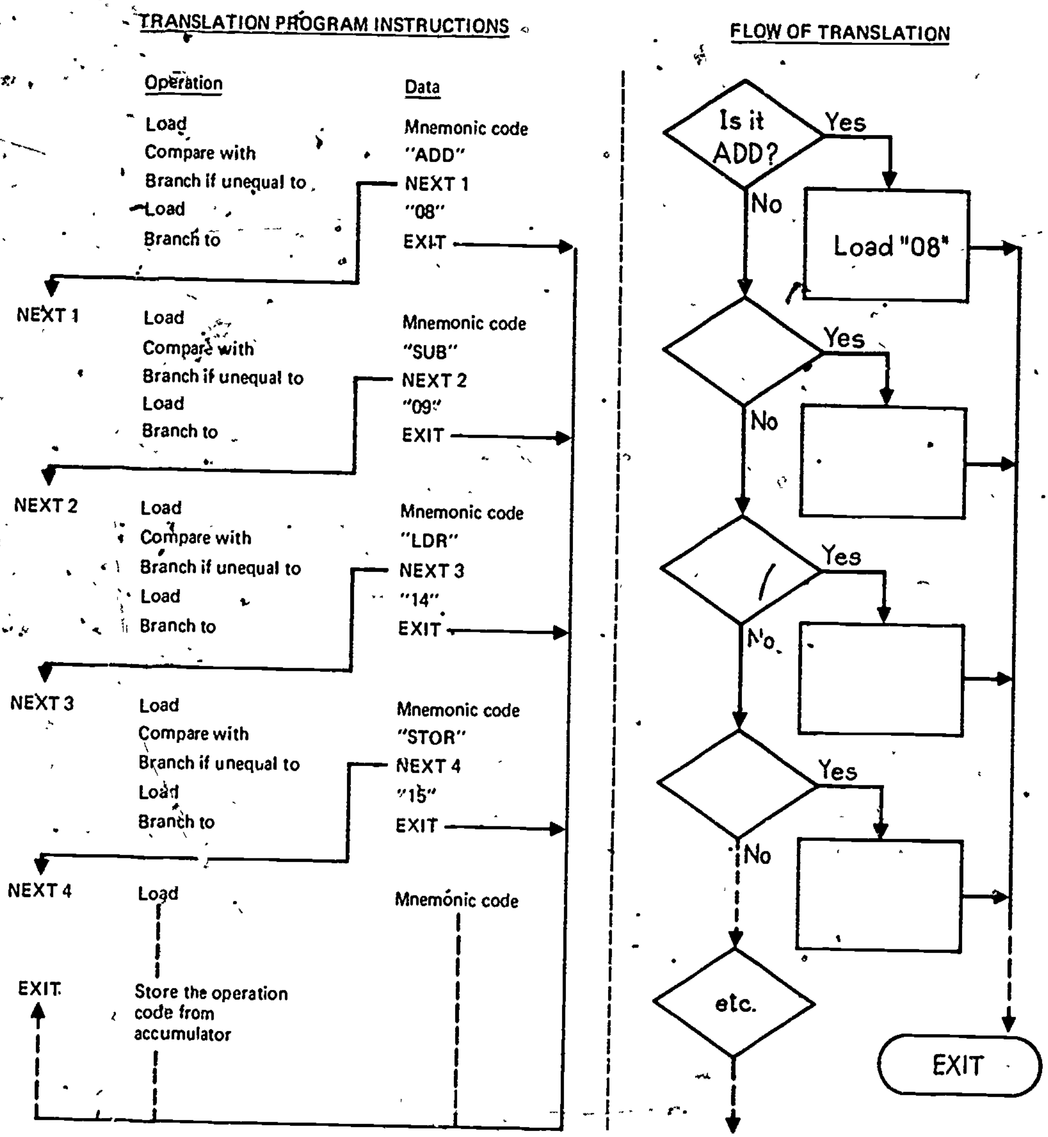


**Fig. 5-32** First part of flow diagram showing translation of mnemonic codes



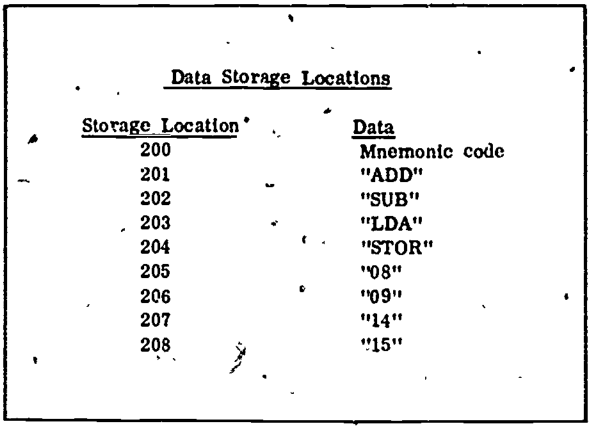
TRANSLATION PROGRAM INSTRUCTIONS

```
        Operation               Data
        Load                    Mnemonic code
        Compare with            "ADD"
        Branch if unequal to    NEXT 1
        Load                    "08"
        Branch to               EXIT
NEXT 1  Load                    Mnemonic code
        Compare with            "SUB"
        Branch if unequal to    NEXT 2
        Load                    "09"
        Branch to               EXIT
NEXT 2  Load                    Mnemonic code
        Compare with            "LDR"
        Branch if unequal to    NEXT 3
        Load                    "14"
        Branch to               EXIT
NEXT 3  Load                    Mnemonic code
        Compare with            "STOR"
        Branch if unequal to    NEXT 4
        Load                    "15"
        Branch to               EXIT
NEXT 4  Load                    Mnemonic code
EXIT    Store the operation
        code from
        accumulator
```

FLOW OF TRANSLATION

Fig. 5–33 Beginning of Mark's draft of the program to translate the mnemonic codes

and 15) so that, when a mnemonic has been identified with ADD, SUB, LDA, or STOR, it can be translated into the correct code 08, 09, 14, or 15. Finally, he would need to assign a location in which to store the



Data Storage Locations

| Storage Location | Data |
|---|---|
| 200 | Mnemonic code |
| 201 | "ADD" |
| 202 | "SUB" |
| 203 | "LDA" |
| 204 | "STOR" |
| 205 | "08" |
| 206 | "09" |
| 207 | "14" |
| 208 | "15" |

**Fig. 5-34** Storage locations assigned to data in program

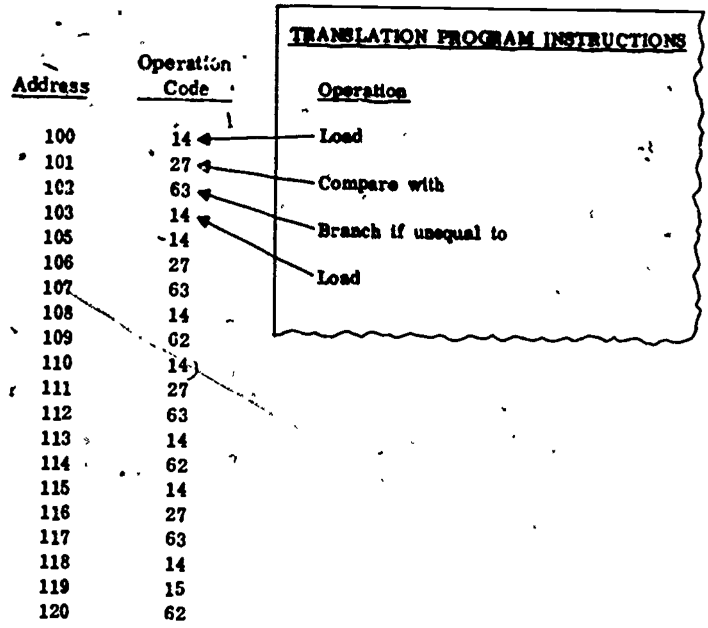
mnemonic to be translated when it is read in, before it is compared with ADD, SUB, LDA, and/or STOR.

Consequently, Mark would proceed now to make a list of these mnemonics and operation codes and to assign them storage location numbers. His list for the first part of his program might look like Fig. 5-34.

At this point, Mark would have everything he needs to complete his programming task. He would know all the operations the Galaxy One could perform, the operation codes for the operations, the assembly language mnemonics for the operations, and the flow that will be involved in the translation process. In addition, he would have his draft of his translator program and a list of all storage locations (addresses) needed to store the data in his program.

Now he just needs to decide where he will store the translator program in the computer's main storage. So, suppose he decided to store the program beginning in location (address) 100, storing one instruction in each consecutive location—100, 101, 102, etc. Then he would go right on to code his program in machine language so it could be loaded into the computer.

His first step would be to write out a list of address numbers (from 100 to, say, 120). Then, following his program one instruction at a time and using one address number for each instruction, he would fill in the appropriate operation code for each instruction in his program. To do this for the program shown in Fig. 5-35, he would refer to the list of operation codes given him in Fig. 5-30 on page 243.



| Address | Operation Code |
|---|---|
| 100 | 14 |
| 101 | 27 |
| 102 | 63 |
| 103 | 14 |
| 105 | 14 |
| 106 | 27 |
| 107 | 63 |
| 108 | 14 |
| 109 | 62 |
| 110 | 14 |
| 111 | 27 |
| 112 | 63 |
| 113 | 14 |
| 114 | 62 |
| 115 | 14 |
| 116 | 27 |
| 117 | 63 |
| 118 | 14 |
| 119 | 15 |
| 120 | 62 |

TRANSLATION PROGRAM INSTRUCTIONS

Operation

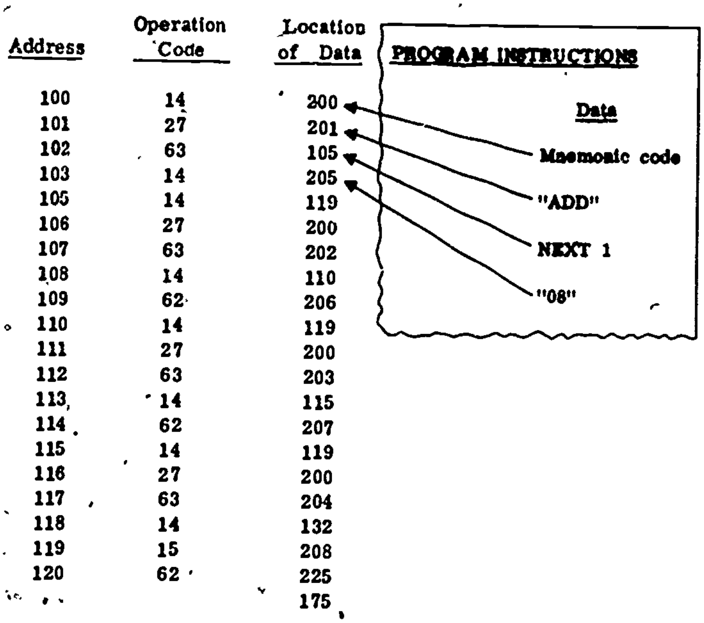
Load

Compare with

Branch if unequal to

Load

Fig. 5-35 Converting program instructions to machine language operation codes



| Address | Operation Code | Location of Data |
|---|---|---|
| 100 | 14 | 200 |
| 101 | 27 | 201 |
| 102 | 63 | 105 |
| 103 | 14 | 205 |
| 105 | 14 | 119 |
| 106 | 27 | 200 |
| 107 | 63 | 202 |
| 108 | 14 | 110 |
| 109 | 62 | 206 |
| 110 | 14 | 119 |
| 111 | 27 | 200 |
| 112 | 63 | 203 |
| 113 | 14 | 115 |
| 114 | 62 | 207 |
| 115 | 14 | 119 |
| 116 | 27 | 200 |
| 117 | 63 | 204 |
| 118 | 14 | 132 |
| 119 | 15 | 208 |
| 120 | 62 | 225 |
|  |  | 175 |

PROGRAM INSTRUCTIONS

Data

Mnemonic code

"ADD"

NEXT 1

"08"

Fig. 5-36 Filling in location of data in program

Next, he would add a column for the location of data and fill in the location for each item of data operated on in the program, as shown in Fig. 5-36.

Once this program is written, it can be loaded into the computer along with the data (and locations) shown in Fig. 5-34 which are used by the program.

Of course, Mark would now have to test and debug this part of the assembler until it worked perfectly. But, when it did, the rest of the Galaxy One software would be programmed in assembly language and Mark's translator would automatically convert it into the Galaxy One's own machine language.

## Check your understanding

1. Which statement below best describes the translator part of the assembler?

   a. It is usually the last program to be developed in the software package.
   b. It is usually the simplest part of the assembler to develop.
   c. It is usually developed by engineers and wired into computers.
   d. It is usually designed to translate high-level languages.

2. The following statements describe the fundamental steps involved in developing the translator part of an assembler. Copy the steps in the logical order in which they would be done.

   Diagram the flow of mnemonic codes and their corresponding machine language operation code.
   Get a list of machine operations and their corresponding machine language codes.
   Draft the program for translating mnemonic codes to machine language.
   Decide on the mnemonic codes to be used in the language for the machine operations.
   Convert the translator program into machine language for loading into the computer.

3. On page 250, you are given a sample program in an assembly language, along with lists of the mnemonics (with corresponding machine language operation codes), and data storage

locations to be used with the program. With the aid of the two lists, translate the program into machine language. Copy the blank machine language program form provided next to the program information to write out your translation. Notice that the address locations for the program are to start at 100.

SAMPLE MNEMONICS AND OP. CODES

| Mnemonics | Operator code |
|---|---|
| LDA | 3 |
| STO | 4 |
| ADD | 1 |
| SUB | 2 |
| PRINT | 5 |
| BR ZERO | 8 |
| BRANCH | 7 |
| HALT | 9 |

SAMPLE PROGRAM - ASSEMBLY LANGUAGE

| | Operation | Data |
|---|---|---|
| | LDA | ZERO |
| | STO | X |
| COUNT | ADD | ONE |
| | STO | X |
| | PRINT | X |
| | SUB | NINE |
| | BR ZERO | END |
| | LDA | X |
| | BRANCH. | COUNT |
| END | HALT | |

SAMPLE DATA STORAGE LOCATIONS

| Location | Data |
|---|---|
| 200 | 000 |
| 201 | 001 |
| 202 | 009 |
| 203 | "X" |

MACHINE LANGUAGE PROGRAM FORM

| Address | Operation | Data |
|---|---|---|
| 100 | | |
| 101 | | |
| 102 | | |
| 103 | | |
| 104 | | |
| 105 | | |
| 106 | | |
| 107 | | |
| 108 | | |
| 109 | | |
| 110 | | |
| 111 | | |
| 112 | | |

*Starting to develop a supervisor.* You can see that creating the very simplest part of the assembler is no simple matter. But, once the whole assembler was written and tested, Mark's team could concentrate on the remaining software. So, let's assume this was done on the Galaxy project and that teams were now assigned to develop the FORTRAN compiler, the COBOL compiler, and the Operating System.

As team leader, Mark was assigned to develop the supervisor. The supervisor routine is, as you know, the most complex and important one in the Operating System. So that you can get just a beginning idea of what would go into its creation, let's look at a very small part of a supervisor that Mark might have developed.

A supervisor usually contains a number of subprograms. Each one of the subprograms performs a specific function. One of the typical functions of a supervisor is to examine an incoming "job" to detect all job control instructions.

A single "job" will usually be entered on punched cards. The first cards in the deck will be "job control cards" which indicate what parts of the operating system will be called on, which translators will be used, and what is to be done. The next cards contain the program, followed by the data cards.

The card deck for a single job can be visualized as shown below in Fig. 5-37.

Interpreting job control instructions could be taken care of by a subprogram which might be called the "job control record analyzer."

The function of this program would be to examine all incoming data to determine if the data record is a job control card. If it is, then the analyzer program would guide the computer in determining which of the available parts of the operating system the card indicates is to be used. The analyzer would then transfer control to the System Loader which would load the desired system or user program and execute it.

In beginning to work out the sequence involved in such an analyzer, Mark might have drawn up a diagram like the one in Fig. 5-38.

Once the flow of operations in this job control record analyzer has been satisfactorily designed, Mark would proceed to write the program for the analyzer in assembly language. Although the diagram of the analyzer in Fig. 5-38 seems only a little complicated, the actual pro-

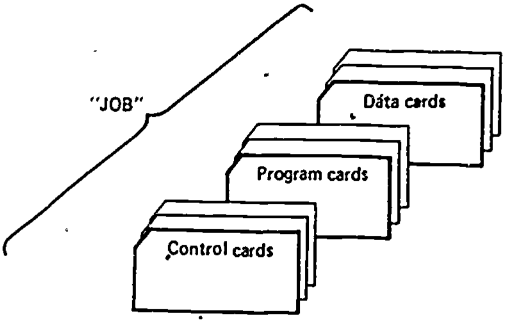


**Fig. 5–37** The cards in a single "job"

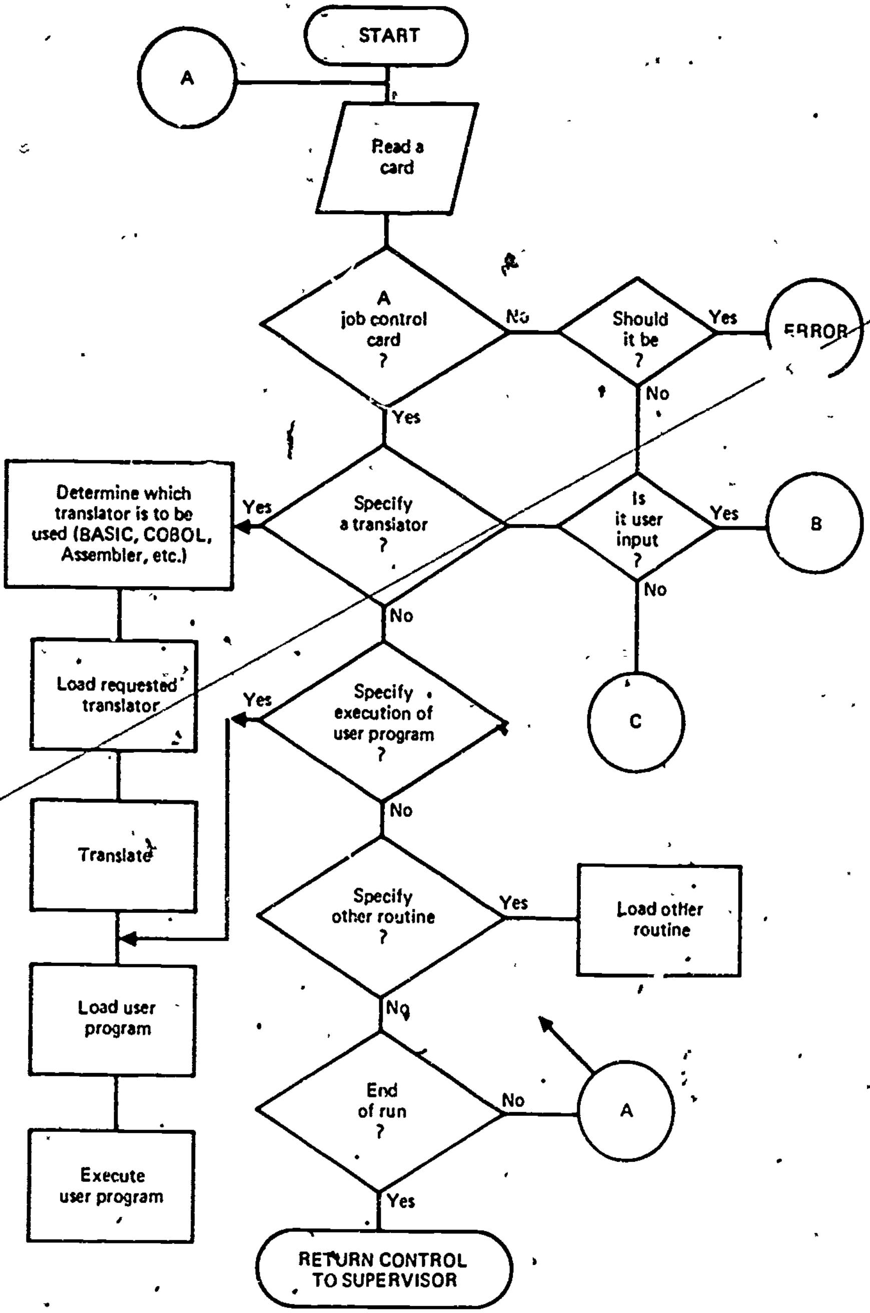


**Fig. 5–38** Diagram showing flow for a job control record analyzer

gramming would be extremely complex. In fact, the program for this quite simple analyzer could easily involve upwards of 1,000 instructions.

Since in this manual you are only learning about the "elements" of systems programming, we will not need to go into the details of the program Mark might have written for the analyzer. But, from a careful study of the diagram in Fig. 5-38, you should have a good understanding of the fundamentals of this small portion of the supervisor.

## Check your understanding

1. A supervisor
   a. is a job control language.
   b. contains several subprograms.
   c. loads the system.
   d. is a translator.
2. Job control cards
   a. tell what parts of the operating system will be used.
   b. come before the program and data in a job deck.
   c. must be read and interpreted by a special subprogram.
   d. all of the above.
3. According to the diagram shown in Fig. 5-38 on page 252 if an incoming job control card does not specify a translator,
   a. the user program is already in machine language and ready for direct execution.
   b. a routine other than a user program is to be loaded.
   c. either one of the above is true.
   d. neither one of the above is true.
4. According to the diagram in Fig. 5-38, the job control record analyzer will return control to the supervisor when
   a. the card being read specifies that a FORTRAN or COBOL compiler is to be used.
   b. a translator needs to be loaded.
   c. a card should be a job control card but isn't.
   d. the end of the run is reached.

*Finishing the Software Package.* The team of systems programmers would, of course, go on to finish the rest of the software for Galaxy One, Two, and Three, including all the I/O programs and utility routines. Then, the BASIC compiler and the time-sharing supervisor for Galaxy Four would be developed. The development of each component of the software would naturally involve its own special problems and complexities. The techniques used in the previous examples would, however, continue to be used as basic tools in developing the system.

Of course, each systems programmer must proceed to test and debug the parts of the software which he or she was responsible for developing. Before the software package is entirely finished, however, it must all be put together and tested as a complete package. The systems programmers will not have finished the developmental part of their job until the entire software system has been tested, debugged, and made to run perfectly in all its phases.

When the software has been completed, the systems programmers must then turn their attention to documenting their work. They now produce both the basic documentation and the complete reference materials for their software components.

Finally, when the reference manuals are ready for use and the Galaxy series of computers has been thoroughly tested and launched, Mark and some of the other systems programmers on his team would probably be available to teach the manufacturer's trainers about the new software. It would then be the responsibility of the trainers to conduct the training sessions for the users of the new Galaxy computers.

Development of a software package often takes years of programming, testing, and revising. Indeed, once the original package is officially issued and customers begin to use it extensively, needed changes always become apparent. Usually, many revisions follow the original issue. The user's systems programmer would be responsible for seeing that the revisions issued by the manufacturer's systems programmer were put into operation on the user's computer system.

Now let's take a look at a day in the life of a user's systems programmer. But first, check your understanding.

## Check your understanding

Following are the basic steps involved in the creation of a software system. Copy the steps in the order in which they usually take place.

Test and debug the software system as a working whole.
Dbcument the components of the system.
Test and debug the individual components of the software system.
Create the several parts of the operating system and all required compilers.
Determine exactly what programs are to be developed for the software package.
Teach trainers about the software system, so they can conduct training sessions for users.

### The user's systems programmer

The typical activities of a user's systems programmer are quite different from those of the systems programmers working for a computer manufacturer. To give you a picture of how the user's systems programmers often spend their time, let's go through an average day with a hypothetical systems programmer, Jack Humphreys.

As you will see, the activities described are much more varied than those of the manufacturer's systems programmer. In the typical day outlined here, the variety and diversity of the work are emphasized. For each of the many problems Jack is confronted with, only the general steps involved in their solution will be described.

Assume that Jack is the lead systems programmer at South State University's Computer Center, where a new ICI Galaxy Four computer has recently been installed. The typical day we will follow is a Monday in the fall of the year.

Jack arrived for work a little tired. The night computer operator had called him at midnight the night before to report a "system crash" —the operating system was not working properly. Jack had spent an hour at the computer center finding and correcting the problem, which turned out to be fairly simple. The problem was that a program was compiled that used too many symbols in each line of instructions and the compiler's symbol table overflowed. Jack fixed the program by reducing the number of symbols used, and the compiler was read into core again from its disk, so that the run could be made.

Jack realized that this "crash" was due to the fact that the compiler did not have an instruction to check for its table being full. Although he knew the system well enough to have often made minor changes in the software himself, the changes needed in this case would be so complex that he decided he would write the manufacturer's field office sometime early this week and ask for a revised compiler.

When Jack arrived at his office this Monday morning, one of the center's three applications programmers was waiting for him. The programmer recently visited a university computer center in another state and learned of an interesting new programming language called "CANADA." She came home with documentation and specifications for the language and wanted Jack to determine how much trouble and cost would be required to implement CANADA on their Galaxy Four. Jack promised her an answer as soon as possible and began to study the documentation.

To decide about the feasibility and cost of implementing CANADA, Jack would first have to determine the level of CANADA as a language and its similarity to existing Galaxy languages. On the basis of these determinations he could decide whether implementation would involve the development of a special compiler or whether he could get by with developing a translator to convert CANADA to another Galaxy language. If both of these alternatives would be too expensive and difficult to implement, he would decide if he could develop a simulator for use with the CANADA language. Such a simulator would be far less efficient in using CANADA, but it would allow the language to be used in the Galaxy Four computer. Once he decided which alternative would be the most feasible, Jack could estimate the cost and time involved in the implementation.

Susan Kelley, the Manager of the Computer Center, soon stopped in to give Jack another job. Until now, the Computer Center had been charging all campus users by the amount of core storage they used and the I/O time used. Starting the first of the following month, Susan wanted the center to begin charging according to both time and priority —higher cost for higher priority. A student or professor at a terminal could request a high priority if the job had to be run immediately and pay accordingly; on the other hand, the student or professor could request a low priority at lower cost and wait awhile for output. Susan asked Jack to build priorities into the operating system so appropriate charges could be made and recorded automatically by the computer every time someone used it. She gave him a set of specifications detailing the new time and I/O charges and the new priority factors to be used.

Jack agreed to have the charging system ready to run by the first of the month. He knew it would take several days to implement and test the new charging system, because it would involve modifying the present job scheduling program as well as the present charging program. The scheduling program would have to be changed to include the priority factors, which he knew would be fairly easy to do with the scheduler now used. Changing the charging program in this case would also be fairly simple because he calculated that he would only need to add a short subroutine to the program.

Jack set aside the folder of information Susan had brought and went back to work on the CANADA language problem. In a short time, the phone rang. It was a graduate student, a user of the Galaxy Four. He had sent some data in to the computer center to be processed using a standard program in the Galaxy Four "Library." The cards had been returned with error messages the computer had printed out. One of the error messages was "program not found." Since the student had used the program many times before with the same control cards, Jack suspected a problem in the Monitor Control Record Analyzer. The student already had called one of the applications programmers for help, but had been referred to Jack. Jack told him to bring the cards and error messages to the computer center and he would check the situation out.

In this case, Jack knew that he would try to rerun the program from the student card deck and obtain a "core dump" printout. He would then have to analyze the printout carefully to determine where the problem was in the Analyzer.

Dr. Van, a professor in the Education Department, was waiting to see him when he finished the telephone call. Dr. Van had just received a government grant of $150,000 to install teletypewriter terminals in 12 high schools. At the moment, all of the terminals connected to the Galaxy Four were ICI 1220's, faster and more expensive than teletypewriters. Dr. Van not only wanted the Galaxy Four adapted for teletypewriter access, but he wanted each school to have its own private access code for logging on the computer. Jack agreed to determine what software changes would be necessary and to order new "interfaces" from ICI to connect the teletypewriters to the Galaxy Four. Jack knew the programming and hardware costs would be paid by the government grant.

After lunch, the Editor of the "Computer Center Newsletter" called to get the details of any recent system changes for publication in the newsletter. This reminded Jack that he had planned to spend the day revising and updating the South State's computer user's manual. He wanted to incorporate into the revised manual all of the changes and updates reported in the newsletter during the past year. Jack had all the newsletters on his desk ready to use, and he figured he could probably write up his draft of the revised manual in just a few hours if he weren't interrupted. So, he closed his office door, set aside the CANADA problem, and spent the afternoon finishing that job.

He was interrupted only once—when Professor Steel called. Professor Steel wanted to use her own access code (73226) to log on the computer, but she wanted to use a program stored by Professor Wickett under a different access code (62483). Jack immediately got on-line to the computer from his office terminal and accessed Professor Wickett's program to insert Professor Steel's access code.

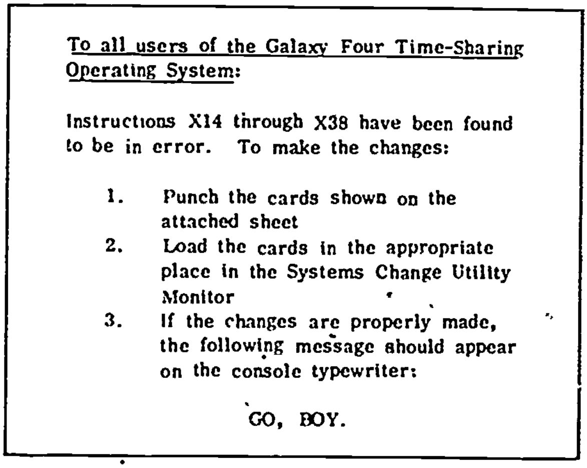
Jack went back to work on the manual revision and had his draft finished by 4:00. He gave the draft to his secretary with instructions to type all the material as soon as possible and return it to him for last minute corrections.

The afternoon mail arrived at 4:30 P.M., and in it Jack found a Systems Notice from ICI. It is shown in Fig. 5-39.

Jack went directly to the data preparation room and had a keypunch operator punch the appropriate cards. He then took the cards and instructions to the computer operator and asked her to load the cards as soon as the present run was finished. She was to let him know if the signal "GO, BOY" did not result.

Jack returned to his office, but before leaving he had three more tasks to complete:

- An applications programmer needed advice on how to get the FORTRAN compiler to compile his program properly, and



To all users of the Galaxy Four Time-Sharing Operating System:

Instructions X14 through X38 have been found to be in error. To make the changes:

1. Punch the cards shown on the attached sheet
2. Load the cards in the appropriate place in the Systems Change Utility Monitor
3. If the changes are properly made, the following message should appear on the console typewriter:

GO, BOY.

Fig. 5–39 An example of a systems notice to computer users

what subroutines were available that he might make use of. It only took a few minutes for Jack to show him how to correct the problem in his program and to advise him about the most useful subroutines.

- Another applications programmer brought in a new program from another computer center. It would replace three separate programs now being used at the University's center. Jack advised her on what job control cards to use to load the new program.
- Last, Jack had to write a notice for the computer center programming staff, announcing the topic for the weekly training session: "The Efficient Use of Galaxy Four Utility Programs."

## Check your understanding

1. Compared to the manufacturer's systems programmer, the user's systems programmer has

   a. much less programming to do on the system software.
   b. much more variety in his or her work.
   c. more short-term tasks to do.
   d. all of the above are true.

2. Which of the following describes what the user's systems programmer usually does when a difficult "bug" is found in the systems software?

   a. Asks the manufacturer to fix the software
   b. Assigns the task of debugging the software to an applications programmer on the user's staff
   c. Revises the software himself or herself
   d. Asks the computer operator to debug the software program

3. Which of the following is typically the responsibility of the user's systems programmer?

   a. Documenting the systems software
   b. Teaching trainers about the software
   c. Helping applications programmers and computer users when they have problems with computer runs
   d. All of the above are responsibilities of the user's systems programmer.

1234567

# Systems analysis and design

## WHAT IS A SYSTEM?

You are a part of dozens of systems. Your family is a system. So is your school. If you have a job, the organization you work for is a system. Within systems, there are many "sub" systems. For instance, in your whole school district (a system) there may be several school buildings, and each building is also a separate system. Within a high-school building, for example, there is a social studies department (a system), a mathematics department (a system), and a main administration office (a system). The school buses operate as part of a transportation system.

What, then, is a "system"? You could probably define it in a general way yourself. A system is a set of "elements" which are related to each other so as to achieve a desired goal. Consider your family as a system. It consists of a set of elements. The elements include people: one or two parents, yourself, perhaps some brothers and sisters. What other elements? Well . . . family pets, a home to live in, perhaps a car, furniture, daily routines for meals and relaxation and chores, various duties and responsibilities assigned to each family member—all the people, devices, and routines that combine to achieve a desired goal. What is the desired goal of your family?

You have probably never considered it, but your family does have a goal, or goals. Perhaps one goal is to raise healthy, happy children. Another may be to achieve some level of economic security. Or perhaps companionship and love and sharing are major goals. Your family, then, is a system. It is a set of elements related so as to achieve a desired goal.

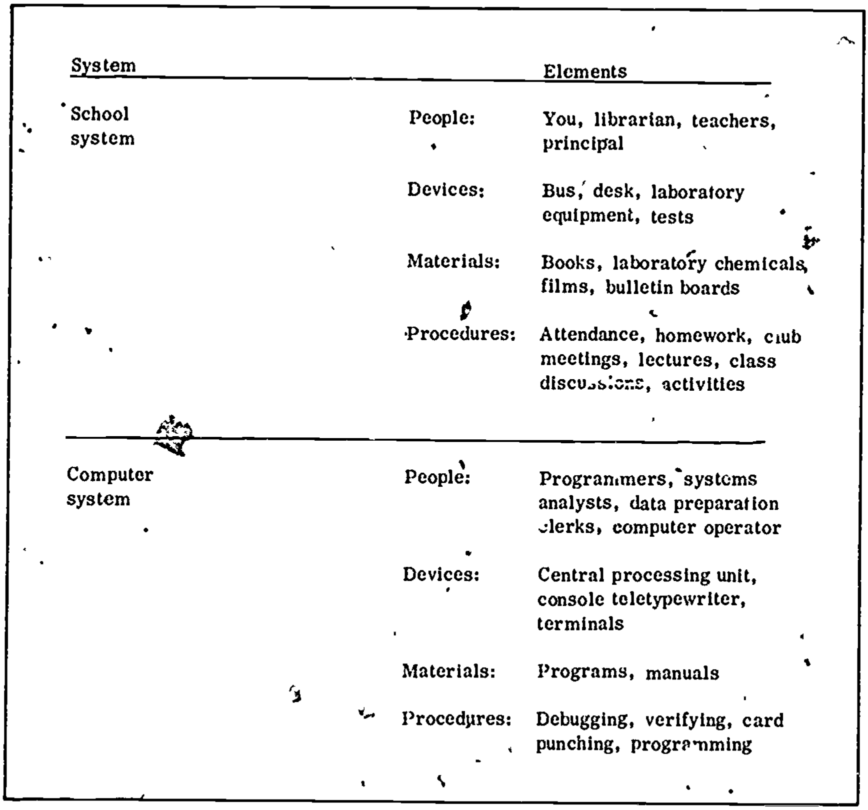
Can you think of the elements of your school which relate to achieve a desired goal? Students. Teachers. Librarians. Custodians. Administrators. Books. Class schedules. Desks, tables, laboratory equipment, films, blackboards, teaching procedures, objectives, workbooks, manuals, even tests! In other words, the elements of the system include *people*, *devices*, and educational *materials* and *procedures*. The desired goal, of course, is to educate, enlighten, and prepare students for a productive and satisfying life. You are an element of the system (the most important element), and so is this book you are reading.

The elements of a computer or data processing system are the procedures and devices (*hardware*), materials in the form of computer programs and manuals (*software*), and the *people* who combine the hardware and software into a well-organized, logical relationship to produce desired results efficiently and economically. People, devices, materials, and procedures, as shown in Fig. 4-1, combine to form a system.

What, then, is "systems analysis and design"? Actually, "analysis" is the opposite of "design." To analyze a system, one breaks down the system into its separate parts or elements and examines the function of each element within the total system, looking for weaknesses or areas in need of improvement. Systems design then begins to combine or build the elements into a new whole. You might compare analysis and design to repairing a bicycle. When you take it apart, tear it down, disassemble the bicycle, you *analyze* the parts and how they relate to the whole. When you have found the problems, you design a way to reassemble the bicycle so it will operate more efficiently.

Do you think you could *analyze* your school and *design* a more efficient system? No doubt you could. Systems are seldom perfect. Analysis and design could conceivably take place in a never-ending cycle to constantly improve systems. Because of the cost, however, most organizations go through systems analysis and design on a periodic basis instead. By examining in detail the tasks performed by various

Fig. 6-1 Systems and elements



| System | | Elements |
|---|---|---|
| School system | People: | You, librarian, teachers, principal |
| | Devices: | Bus, desk, laboratory equipment, tests |
| | Materials: | Books, laboratory chemicals, films, bulletin boards |
| | Procedures: | Attendance, homework, club meetings, lectures, class discussions, activities |
| Computer system | People: | Programmers, systems analysts, data preparation clerks, computer operator |
| | Devices: | Central processing unit, console teletypewriter, terminals |
| | Materials: | Programs, manuals |
| | Procedures: | Debugging, verifying, card punching, programming |

people and the flow of data into and out of the system, better systems for achieving the organization's goal can be designed. This kind of system analysis and design even takes place in families. Probably you have been involved in rearranging the furniture, reassigning household chores, or developing new family rules in order to achieve mutual goals more effectively or with less strain on family members.

Frequently when an organization undertakes a systems analysis and design, it investigates the desirability of using a computer to help achieve some of its goals. Because acquiring a computer might require a substantial investment, this aspect of the systems analysis is usually scrutinized very closely—by doing a *feasibility* study.

Sometimes a thorough feasibility study results in streamlining of current manual methods of processing data, so that a computer is not needed at all. Other times, the recommendation is for the acquisition of a computer system or purchasing time on someone else's computer. Before making such a recommendation, however, the people doing the feasibility study will have looked for ways in which the computer could help reduce costs, increase profits, and/or project a more favorable, progressive image to the public. These benefits must be balanced against the actual costs of providing the computer services, including such things as hardware, software, personnel, supplies, and space. At the same time, the study team will look for areas in which computerization can improve the operation of the organization. These areas usually possess one or more of the following elements:

- *High volume* of transactions or information: receipts, sales orders, purchase orders, and inventory data are examples of the large number of different types of information which might be necessary for only one area.
- *Repetitive* type of transactions: frequent repetition of these high volume transactions.
- *Common source documents:* a single document is used for several purposes. For instance, a sales order might be used for shipping the item, billing the customer, and crediting the salesperson for commission.
- *Mathematical* processing: a great deal of computation is required.
- Need for *quick responses:* a report is needed immediately, as soon as the input data is available. An example of this would be a sales commission statement needed the day after a work week ends.

As you can see, such a feasibility study is actually a type of systems analysis, bent on finding out if it would be feasible to install or to purchase time on a computer.

If the decision is made to computerize, the organization will then need, probably, the full-time services of one or more systems analysts. This position is one of the key links in the communication chain between humans and computers. Broadly speaking, the analyst is the person who analyzes the data processing needs of his or her own organization or of outside clients. Using the elements available, the systems analyst designs complete systems to fill those needs. Chapter 7, in its discussion of computer career opportunities, describes the role and responsibilities of the systems analyst in more detail.

## Check your understanding

1. A system is a set of (?) which are related so as to (?).
2. Systems analysis
   a. separates into individual parts.
   b. separates into parts, then recombines into a whole.
   c. puts together a lot of pieces to form a whole.
   d. relates the parts to achieve a desired goal.
3. Systems design
   a. is the design of a system.
   b. usually follows systems analysis.
   c. disassembles systems.
   d. both a and b.
   e. both a and c.
4. A feasibility study
   a. always results in a recommendation for computerization.
   b. is a type of systems analysis.
   c. is a type of systems design.
   d. is done to determine the feasibility of hiring a systems analyst.
5. A systems analyst
   a. analyzes systems.
   b. designs systems.
   c. analyzes data processing needs.
   d. both b and c.

## THE "SYSTEMS APPROACH"

Let's assume that you have (or hope to develop) the logical and imaginative mind, the talent for communication, and the programming and computer operation skills that would make you a good candidate for a system analyst's career. How would you go about master-minding the solution of one of those super-problems?

It might be tempting at first to think of just jumping into the middle with ideas and, by trial and error, putting together a solution that would work. In fact, if the problem were fairly simple and well-defined, your expertise would probably make it easy for you to come up with some solutions this way. But, remember, the problems systems analysts are given to solve are usually broad-based and are often presented in fairly vague terms. Without a systematic approach you could jump in and never be heard from again.

To steer their way through the many-faceted problems so common in the work, systems analysts use a procedure called the systems approach. You probably already know the basic steps in this approach because they are the classical five steps in scientific problem-solving:

1. Define the problem.
2. Gather and analyze data.
3. Develop alternate solutions.
4. Decide on the best solution.
5. Put the decision into effective action.

These basic steps, in fact. can be used in almost any problem situation we might run into.

In systems analysis, these steps are translated into terms which suit the kinds of problems involved. A typical systems approach outline based on these five steps is

1. *Define* all the output needed to achieve the goals of the system. (This is the most essential, and generally the most difficult, step in the entire process.)
2. *Gather and analyze* data to determine both the fixed and the variable elements of input. (Fixed elements are the unchanging elements, such as name, age, and social security number. Variable elements are those elements which might change, such as salary, grade point average, etc.)

3. *Develop* (design) the system so the variable elements of input and the process produce the desired output as efficiently as possible.
4. *Test* the system and make needed corrections and revisions on the basis of time, cost, or effort factors.
5. *Put the system into effect by*
   a. documenting it (that is, writing detailed procedures for all input, output, and processes involved), and
   b. implementing it.

Notice how closely the "systems approach" steps parallel the scientific problem-solving steps listed before.

Later, we will follow the step-by-step solution of a systems problem using the systems approach. First, however, let's look more closely at these steps to see what each one actually involves.

## Defining output/input and analyzing data

Defining the output and input of a system, of course, is the most critical step in the process. The contents and types of output should always be determined first. Then the analyst goes on to determine what elements of input are necessary to produce the output desired.

To determine the contents of both output and input, the analyst must work very closely with the client. Since the client may not have an altogether clear or even complete idea of needs and options, it is up to the analyst to analyze the situation carefully enough with the client that all input are finally accurately defined and the output needs are correctly and completely assessed.

For example, the owner of a small plastics company might decide he needs to use data processing to handle his payroll. He may, however, have the idea that all he really needs is just a system that will print out payroll checks.

It would be up to the systems analyst to ask the owner about all the other possible outputs which might be useful or necessary to his payroll department. Such outputs might be items like file copies of all checks, complete lists of each payroll, monthly summary statements, departmental break-downs of payroll for budget adjustment, and so forth. Ultima'ely, the systems analyst would need to decide which outputs are feasible and worthwhile, given the owner's needs.

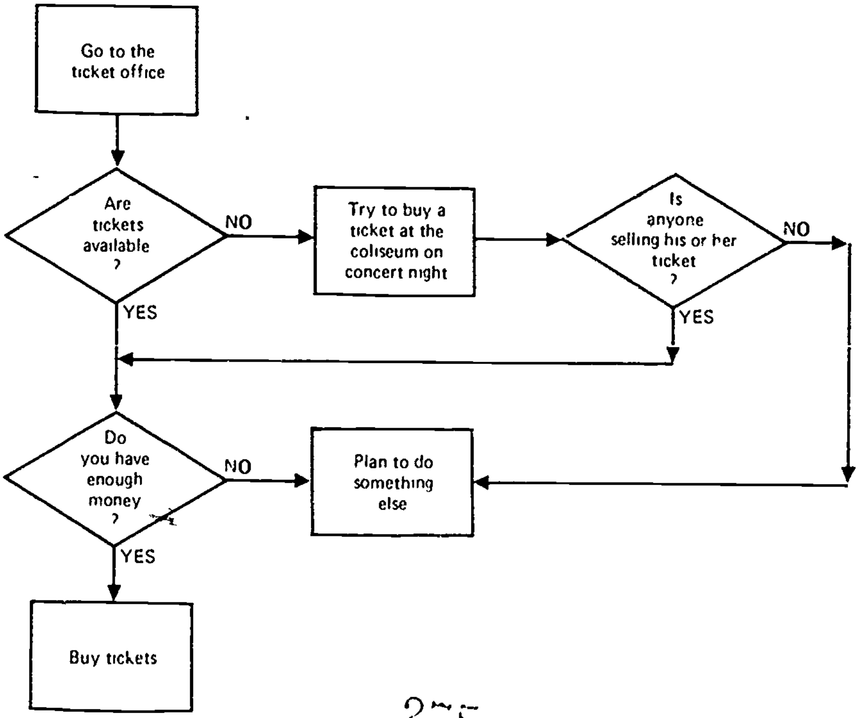
Further, the owner might assume the only input he has available is that of the salary figures and employee identification. On closer analysis, however, the systems analyst might discover that such additional information as different hourly rates, actual hours worked, overtime hours, complete deduction data, and budget allocations are all available as input data for the required system. Once all the input has been gathered, it would be up to the systems analyst to determine the exact fixed and variable inputs to be used by the system.

As you can see, the first two steps of the systems approach require the analyst to rely both on imagination when it comes to analyzing possibilities and on practicality when it comes to making final decisions.

## Designing the system

To help them carry out the third step in the systems approach—that of developing a new data processing system—systems analysts

**Fig. 6–2** Flowchart for buying a concert ticket


Go to the ticket office
Are tickets available ?
NO
YES
Try to buy a ticket at the coliseum on concert night
Is anyone selling his or her ticket ?
NO
YES
Do you have enough money ?
NO
YES
Plan to do something else
Buy tickets

commonly use three special tools. These are block diagrams, systems flowcharts, and decision tables. These tools are of considerable help in gathering details together, analyzing complex situations, and outlining the system's flow. Let's consider them one at a time.

You already may be familiar with flowcharts which simply illustrate the steps needed to accomplish certain acts. For example, you could flowchart the sequence of events in buying tickets for a rock concert at the coliseum as shown in Fig. 6-2.

A computer programmer uses a different form of flowchart, called a micro-flowchart, to help organize the steps in a program. A very simple example is shown in Fig. 6-3.

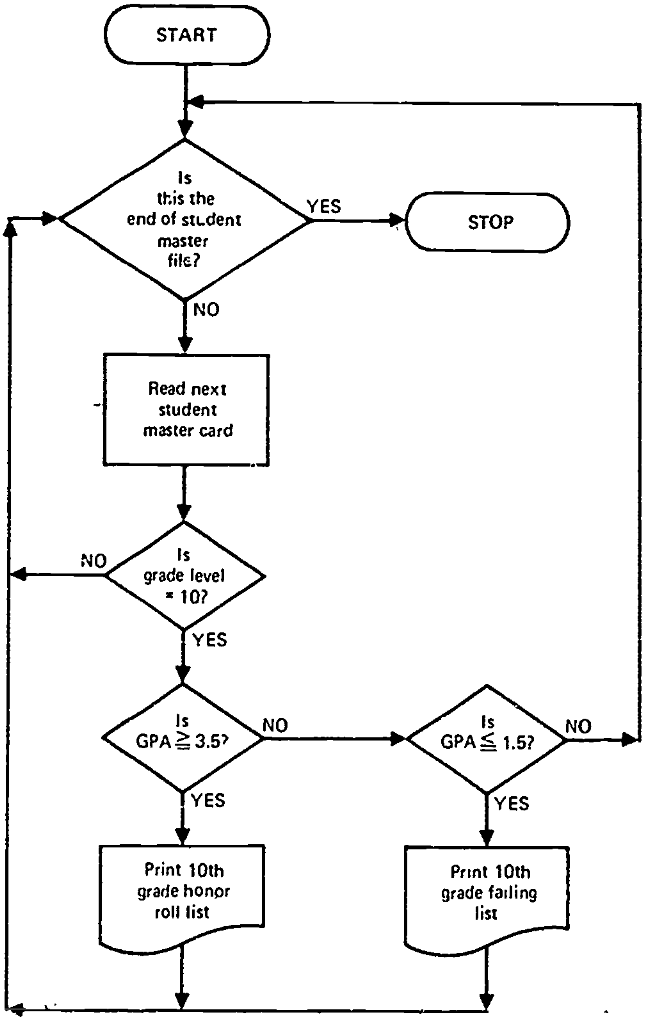

START
Is this the end of student master file?
YES
STOP
NO
Read next student master card
Is grade level = 10?
NO
YES
Is GPA ≧ 3.5?
NO
YES
Is GPA ≦ 1.5?
NO
YES
Print 10th grade honor roll list
Print 10th grade failing list

**Fig. 6-3** A programmer's flowchart

You can see from this example that the program flowchart, or micro-flowchart, details the sequence of steps, one by one, needed to progress from one state (condition) to the next toward the goal. Since the programmer must give such step-by-step instructions to the computer, a flowchart showing each small ("micro") detail is needed.

In contrast to the programmer, however, the systems analyst deals with problems on a more general level. A programmer, for example, may develop a specific program to make the computer compute grade point averages and print out an honor-roll list. The systems analyst might design an entire system in which student personal records, class and grades information, national test scores, attendance records, and other data can be input and processed to produce updated report cards, honor-roll lists, class lists, school directories, state reports, and other output needed. After such a system is designed, the analyst works

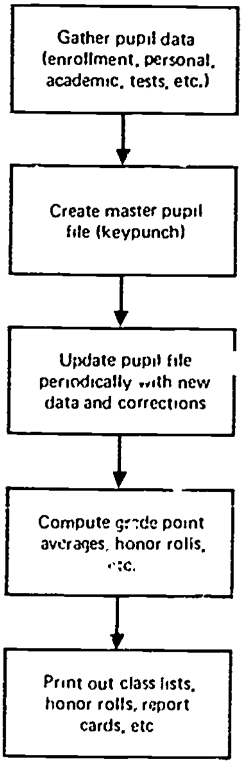

Gather pupil data (enrollment, personal, academic, tests, etc.)
Create master pupil file (keypunch)
Update pupil file periodically with new data and corrections
Compute grade point averages, honor rolls, etc.
Print out class lists, honor rolls, report cards, etc

Fig. 6–4 A block diagram

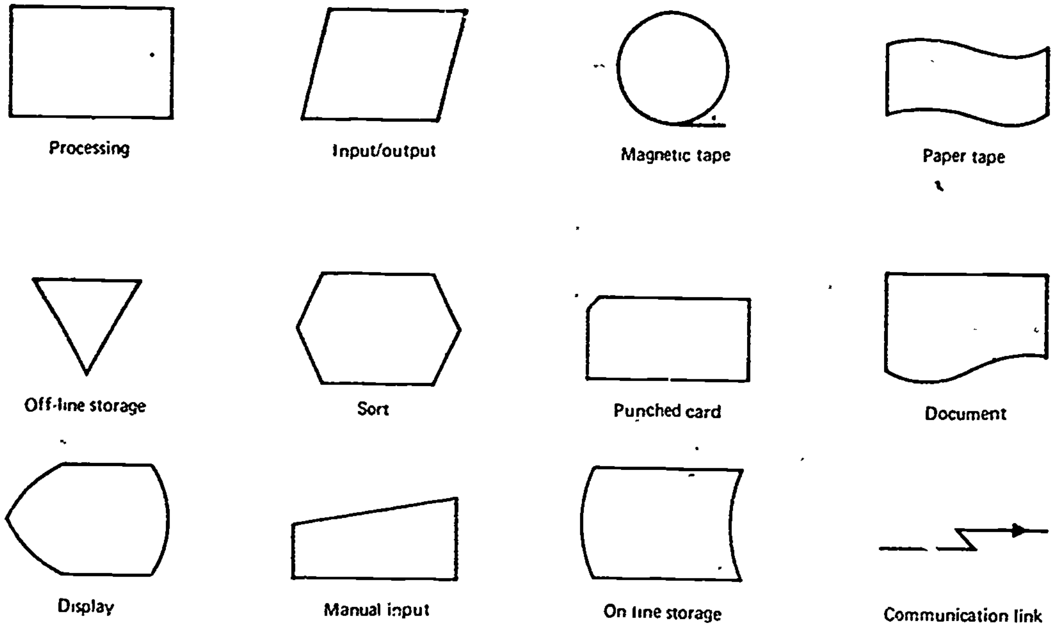


Fig. 6–5 System analyst's flowcharting symbols

closely with one or more programmers who will write the detailed programs needed to process the data and produce the reports.

*Block diagrams.* To flowchart such broad-based problems the systems analyst often starts with what is called a block diagram. The block diagram serves to break a large problem down into its general sections and to show them in logical sequence. Fig. 6-4 is an example of a block diagram. You should be able to read this diagram easily and to gather from it a general picture of the problem it represents.

*Systems flowcharts.* Once such a general picture of the problem is established, the analyst begins to refine the picture. The blocked-out steps are more closely analyzed, and they are broken down into clearer and more precise steps. In the process, the analyst is very likely to use a flowchart something like the programmer's to graphically illustrate the flow of specific major steps involved in the system being developed.

The symbols the systems analysts most commonly use for their flowcharts are shown in Fig. 6-5. You will notice that several of these symbols are general ones found in *program* flowcharts as well as *sys-*

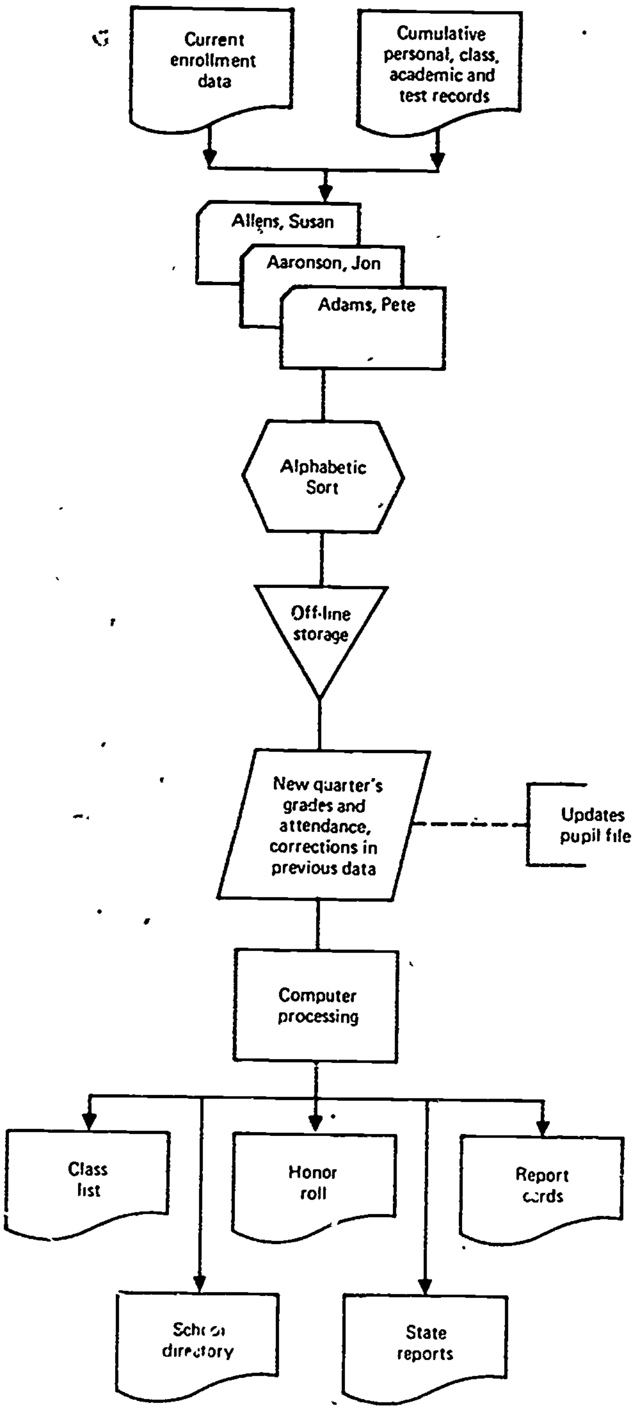


**Fig. 6–6** A system analyst's flowchart

*tems* flowcharts. The additional symbols pictured in Fig. 6-5 help illustrate the different purpose of the systems flowchart. The program flowchart shows details oniy of the CPU processing, while the systems flowchart exists to diagram the flow of logic and data through the system from conception to finish. The systems analyst uses special symbols, therefore, to indicate the manner in which data flow from one step to the next—for instance, in what form they were submitted, how they are to be fed into the computer, what kind of storage is needed, whether a printout is to be made, and so on. Figure 6-6 is an example of a flowchart a systems analyst might construct to diagram a system for solving the problem discussed earlier.

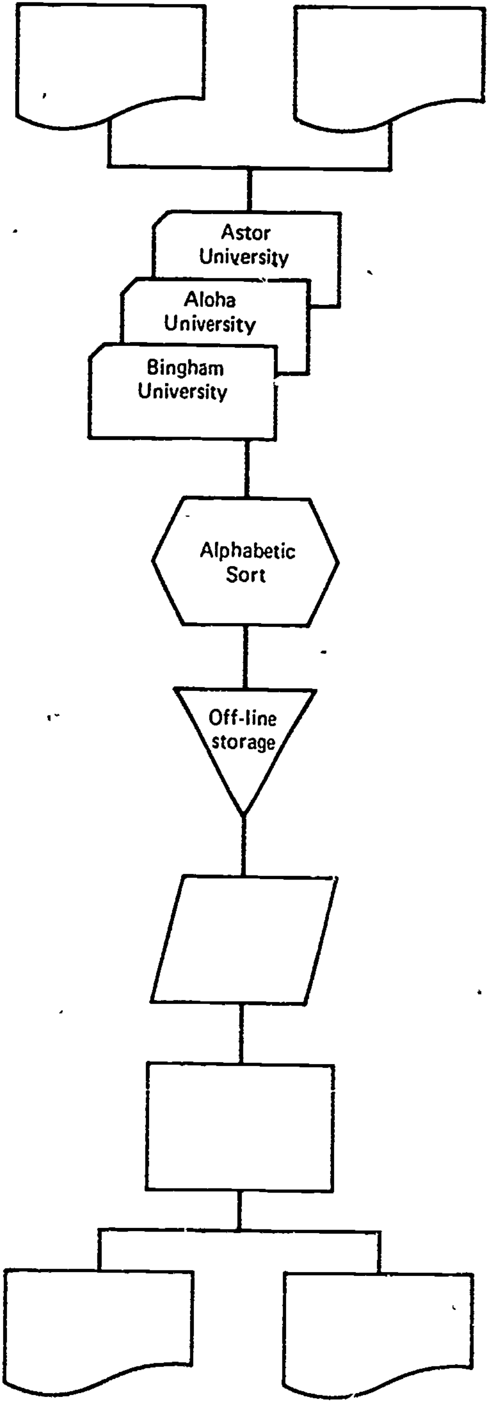
Study this flowchart for a few minut s. Once you have identified the meaning for each of the symbols, the chart should be easy to follow. Can you see how this flowchart would give the analyst a clearer picture of the major steps involved in this system than the block diagram provided in Fig. 6-4? With such a picture at hand, the analyst can proceed to work out the specifications for each step in the process. Because analysts' flowcharts focus on the specific major steps in the overall picture, they are usually called "macro" (or "large") flowcharts. To get a more vivid idea of how much "larger" their picture is than that of a programmer's flowchart, notice the operation box marked "Computer Processing" in Fig. 6-6. This box shows where all of the programming operations take place. To fill in the computer programs needed to make this system work, it might take the programmer a dozen pages of flowcharts, but they would all fit into the macro-flowchart as just a part of this one box marked "Computer Processing."

When a system has been completely enough diagrammed in macro-flowchart form, the analyst can give this chart to the programmer to show generally what programs will be called for in the system. On the basis of the macro-flowchart and any additional specifications needed, the programmer can create the programs for the system.

## Check your understanding

1. The "systems approach" is

   a. a way of flowcharting problems.
   b. based on the classical five steps in scientific problem solving.
   c. a substitute for flowcharting.
   d. the last step in developing a system.


Astor University
Aloha University
Bingham University
Alphabetic Sort
Off-line storage

2. Assume you were about to design a system to help students analyze information about colleges and decide which ones they should apply to for admission. Assume that so far you have outlined the following steps in your approach to the problem:

   Document the procedures for running the system.

   The input will consist of 20 kinds of data for universities (such as names, locations, tuition figures, departments established, etc.) taken from university catalogs or university bulletins.

   Test out the final system and make any revisions that are needed.

   Two kinds of output will be sufficient:

   a. a list of all the universities which satisfy the conditions specified by the user (e.g., all universities with tuitions under $200 per semester and a major department in Black Studies), and
   b. for any university, a list of items the user is interested in (e.g., at a given university, a list of all scholarships available).

   Design the system to process the input to produce the desired output.

   Write these steps in the order in which you would be working on them if you used a traditional systems approach.

3. Assume you are still working on the problem given in question 2. On a separate sheet of paper, draw the macro-flowchart shown on page 274 and fill it in to illustrate the basics of the system you would need to design.

***Decision tables.*** Sometimes the problems the systems analyst must solve are too complex to be translated immediately into a macro-flowchart. Often, for example, the input and output have so many facets it is difficult at first to determine exactly how they should be shown to flow on a flowchart. In these cases, decision tables are an invaluable tool to the analyst.

In general, a decision table may be defined as a table showing (by X's) what actions will be taken when any combination of conditions

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| **Conditions** | | | | |
| Does the shop have 10-speed bikes in stock? | Y | Y | N | N |
| Do you have enough money? | Y | N | Y | N |
| **Actions** | | | | |
| Buy a 10-speed bike. | X | | | |
| Order a 10-speed bike from their next shipment. | | X | X | X |
| Save more money. | | X | | X |

Fig. 6–7 Decision table for buying a bike

holds (signified by Y's and N's, for "yes" and "no"). Here is a very simple example. Can you make sense of it?

If you had trouble reading the table, here's how to do it. First, read the two basic conditions in this problem and the three possible actions to be taken. Now, look at the situation in column 1. The Y's (or yes's) mean that, in this case, the shop has 10-speed bikes and you have enough money. So, the appropriate action in this case (shown by X) is buy a 10-speed bike. Column 2 shows the situation where the shop has 10-speeds (Y), but you haven't enough money (N). The actions to be taken are order a 10-speed bike and save more money. Column 3 shows that in the situation where the shop has no 10-speeds, but you do have enough money, the only action to be taken is to order a 10-speed from the next shipment. Read down column 4 in this way. Notice that, for this problem, the four columns cover all possible situations.

The vertical columns on a decision table may also be called "rules." If you look again at the table in Fig. 6-7, you can see why. Column 1 can be read: "If the shop has 10-speeds and you have enough money, then you can buy a 10-speed." This is phrased as a simple "if-then" rule. In the same way, the next column can be read: "Rule 2—If the shop has 10-speeds, but you don't have enough money, then you can order a 10-speed and save more money." Read columns 3 and 4 as rules.

You have probably already decided it would be just as easy to flowchart the problem about 10-speed bikes. Right! This is a very simple example used just to show you how easy it is to read a decision table. Now, let's turn to a more complex problem—one that the beginning systems analyst might find hard to flowchart right off the bat! It is in such situations that the analyst will use a decision table to clarify all the possibilities in the situation before constructing a systems flowchart.

Suppose a small airline company wants to start automating its ticketing procedures. Assume the manager gives you the following information about how requests for tickets are to be handled.

If a first-class request is received, and

a. first-class is available for the flight, then a first-class ticket is issued and one is subtracted from the first-class seats available for the flight.
b. first-class is not available, then tourist-class is checked. If tourist-class is open and an alternate class is acceptable to the customer, then a tourist-class ticket is issued and one is subtracted from the tourist-class seats available.
c. an alternate class is acceptable, but neither first-class nor tourist-class is open, then the customer is placed on the wait lists for both classes.
d. an alternate class is not acceptable, but first-class is not available, then the customer is placed on the wait list for first-class only.

If a tourist-class request is received, and

a. the tourist-class is available for the flight, then a tourist ticket is issued and one is subtracted from the tourist-class tickets available.
b. tourist-class is not open, then the first-class is checked. If the first-class is open and an alternate class is acceptable to the customer, then a first-class ticket is issued and one is subtracted from the first-class tickets available.
c. an alternate class is acceptable, but neither the tourist-class nor first-class is available, then the customer is placed on the wait lists for both classes.
d. an alternate class is not acceptable but tourist-class is not available, then the customer is placed on the wait list for tourist-class only.

Do you think you could flowchart this situation from this bulky set of rules? If so, try it and see how you do. But for those who think it might be very difficult to organize the details in their minds, let's see how this problem would look if it were set up in a decision table.

First, to construct the table, we need to know all the conditions (or "if's") that are possible in the situation, and all the actions (or "then's"). Reading through the rules the airlines provided, we can list the conditions and actions to start the table as shown in Fig. 6-8.

| Conditions | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Is request for first-class? | | | | | | | | |
| Is first-class open? | | | | | | | | |
| Is tourist-class open? | | | | | | | | |
| Is alternate class acceptable? | | | | | | | | |
| Is request for tourist-class? | | | | | | | | |
| **Actions** | | | | | | | | |
| Issue first-class tickets. | | | | | | | | |
| Subtract one from first-class tickets available. | | | | | | | | |
| Issue tourist-class ticket. | | | | | | | | |
| Subtract one from tourist-class tickets available. | | | | | | | | |
| Place on first-class wait list. | | | | | | | | |
| Place on tourist-class wait list. | | | | | | | | |

**Fig. 6–8** First step in outlining airline ticketing decision table

| Conditions | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Is request for first-class? | Y | Y | Y | Y | | | | |
| Is request for tourist-class? | | | | | Y | Y | Y | Y |
| Is first-class open? | Y | N | N | N | | Y | N | |
| Is tourist-class open? | | Y | N | | Y | N | N | N |
| Is alternate class acceptable? | | Y | Y | N | | Y | Y | N |
| **Actions** | | | | | | | | |
| Issue first-class ticket. | X | | | | | X | | |
| Issue tourist-class ticket. | | X | | | X | | X | |
| Subtract one from first-class available. | X | | | | | X | | |
| Subtract one from tourist-class available. | | X | | | X | | | |
| Place on tourist wait-list. | | | X | | | | X | X |
| Place on first-class wait list. | | | X | X | | | X | X |

**Fig. 6–9** Airline ticketing decision table

Now, referring again to the set of rules provided before, we can fill in the columns to show what the if-then rules are for each possible situation. The completed table would look like the one shown in Fig. 6-9. Notice that the order of the conditions listed in this table is different from that in Fig. 6-8. The sequence was changed to put the conditions in the most useful order. Do you see any errors in columns 7 and 8?

On the basis of this decision table, a macro-flowchart will be much

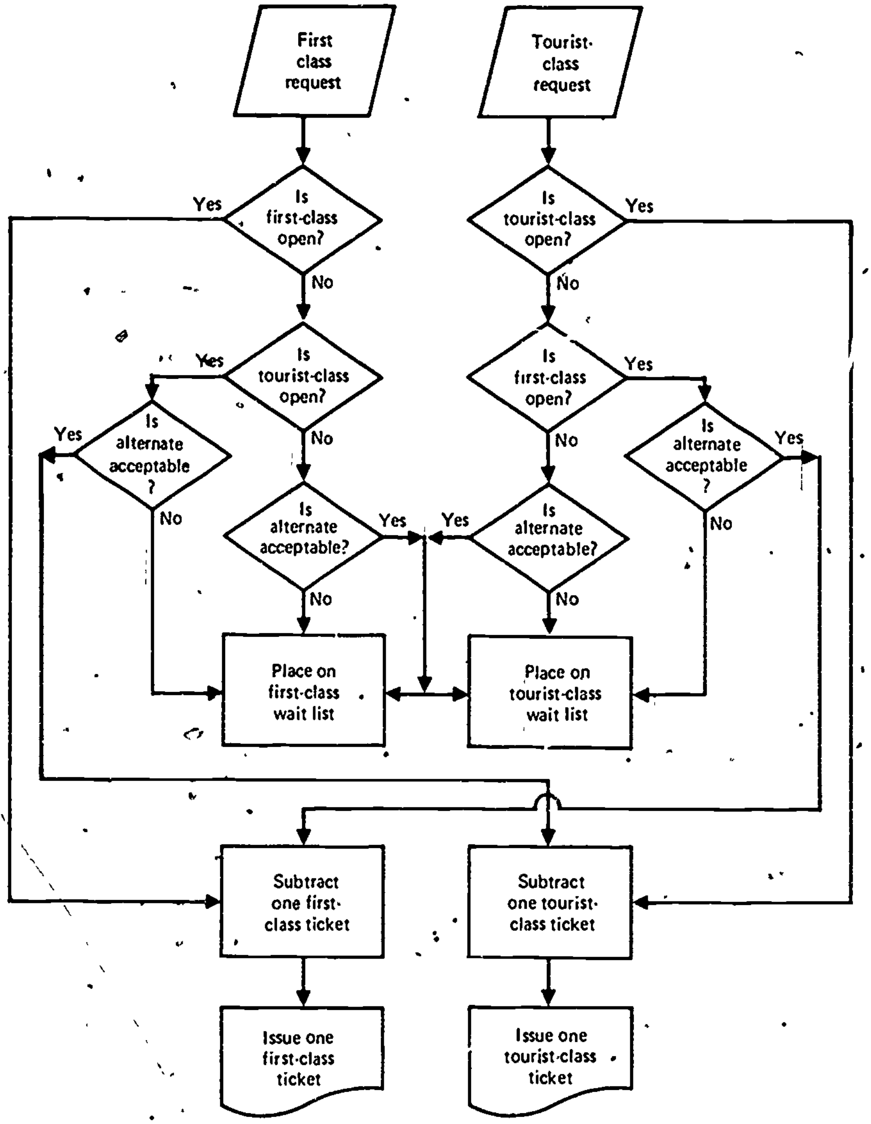

First class request
Tourist-class request
Is first-class open?
Yes
No
Is tourist-class open?
Yes
No
Is tourist-class open?
Yes
No
Is first-class open?
Yes
No
Is alternate acceptable ?
Yes
No
Is alternate acceptable?
Yes
No
Is alternate acceptable?
Yes
No
Is alternate acceptable ?
Yes
No
Place on first-class wait list
Place on tourist-class wait list
Subtract one first-class ticket
Subtract one tourist-class ticket
Issue one first-class ticket
Issue one tourist-class ticket

**Fig. 6-10** Macro-flowchart of airline ticketing problem

easier to construct, step-by-step, and it will be easier to doublecheck to see that no alternative or rule has been left out. A sample of one way to flowchart the ticketing problem is shown in Fig. 6-10. As you study this macro-flowchart, notice that there is no "computer processing" box. In this case, the major steps in the processing have been diagrammed on the basis of the decision table. Note also, however, that only the major steps have been accounted for. The programmer will still have to break down this area of the flowchart into micro-flowcharts.

# Check your understanding

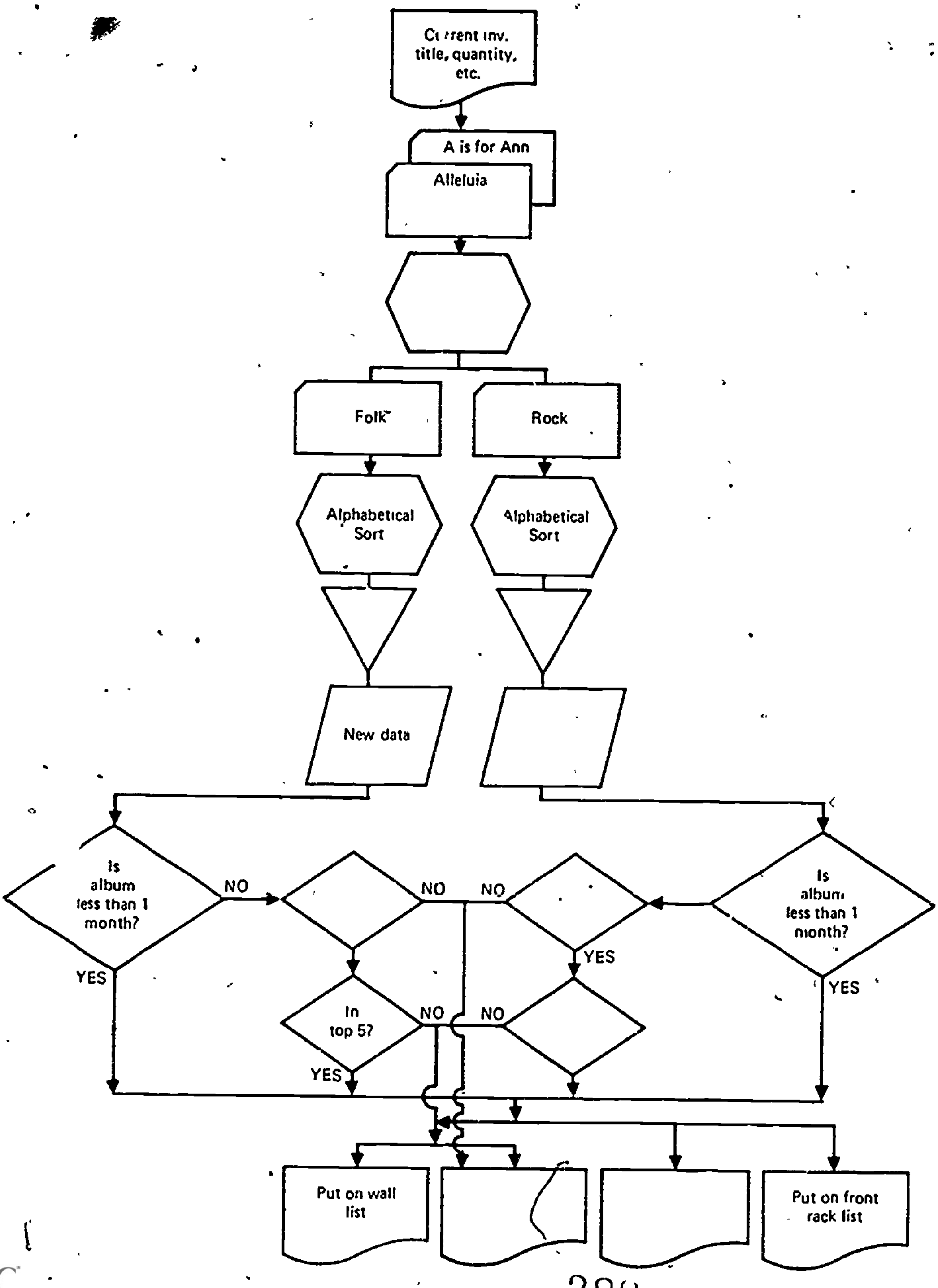
1. Micro-flowcharts and macro-flowcharts
    a. are both used by systems analysts to solve little and big problems.
    b. differ in name only.
    c. differ in the kind and amount of detail they show.
    d. are used by junior and senior analysts, respectively.
2. Decision tables are most useful in organizing information when problems are
    a. too simple to flowchart.
    b. too simple to matter.
    c. too simple to believe.
    d. none of the above.
3. Assume that a folk and rock record store owner has given you the following problem:

    I want a system that will give me not only a weekly list of albums I need to order, but one that will tell me where to display the albums I have. I figure I should display the five best-selling rock albums and the five best-selling folk albums in the front windows, as well as file them in the front windows, front racks, and regular stacks, and on the wall. The top 15 folk sellers and the top 15 rock sellers should be displayed on the store walls and should be filed-in the regular stacks. (Any others will automatically be in the stacks alone.) I want a new display and file list every week.

    a. On a separate sheet of paper, draw the decision table below and complete it for the "display" part of the problem.

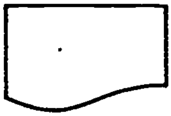
| Conditions | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Is album one of 17 top selling folk albums? | Y | Y | Y | Y | N | N | | | | | | |
| Is it one of 5 top folk albums? | Y | Y | N | N | N | N | | | | | | |
| Is album one of 15 top selling rock albums? | | | | | | | Y | Y | Y | Y | N | N |
| Is it one of 5 top rock albums? | | | | | | | Y | Y | N | N | N | N |
| Is less than one month old? | Y | N | Y | N | Y | N | Y | N | Y | N | Y | N |
| Actions | | | | | | | | | | | | |
| Display in windows. | | X | X | | X | | | | | | | |
| Display on walls. | | X | X | | | | | | | | | |
| File in front racks. | | X | X | | X | | | | | | | |
| File in regular stacks. | | X | X | | X | | | | | | | |

b. Draw the flowchart below and complete it to show a system that could give the store owner the output he wants.



## Designing the input forms

In the work you have done so far in designing systems, you have been using the symbol



for both input and output documents. You probably have a fairly clear idea of the distinction between the two. The input (or "source") documents are just those which present the data needed for processing. The output documents are those produced by the system. Incidentally, it should be clear to you that the output documents are really the "payoff" of the entire system.

All the computer processes that take place between input of documents and the output of new documents are primarily the concern of the programmer and the systems analyst. However, the client (or system user) shares a concern for the documents with the analyst. It is, after all, the user who will be preparing the input documents and who will be using the documents output by the system. Whenever a data processing system is to be developed, the analyst always works very closely with the client on both the content and the form these documents will have. The definition of the contents of input and output is usually done at the beginning of a systems analysis. You have already studied this aspect of the documents on pages 267 and 268.

The second aspect of input and output documents is their form. These can vary widely and the analyst is often responsible for designing them at the same time that the rest of the system is being designed. So, you should be familiar with how these forms are designed. First, we'll look at the possible forms of input documents and some criteria the analyst can use in deciding on the ones to use. Then, we'll consider the forms output documents may take. For both kinds of forms, however, the most important consideration is always that the information be presented in as usable a way as possible.

The form of input documents (which carry the handwritten data from the user to the computer) depends first of all on the type of hardware used. Where a mark or optical reading device is used for inputting, the data are marked on forms which go directly into the reading device.

In this case, the forms are designed with both the user's convenience and the device's capabilities in mind.

One commonly used mark-sense form is the standard IBM test answer sheet. You have probably used this form on national or state tests. On it you marked each answer by blacking out the appropriate small column with a pencil.

The data may not lend itself to this form, or to any other standard mark-sense form, however. Then the systems analyst will have to design the mark-sense form or forms which will be suitable for the data and readable by the device.

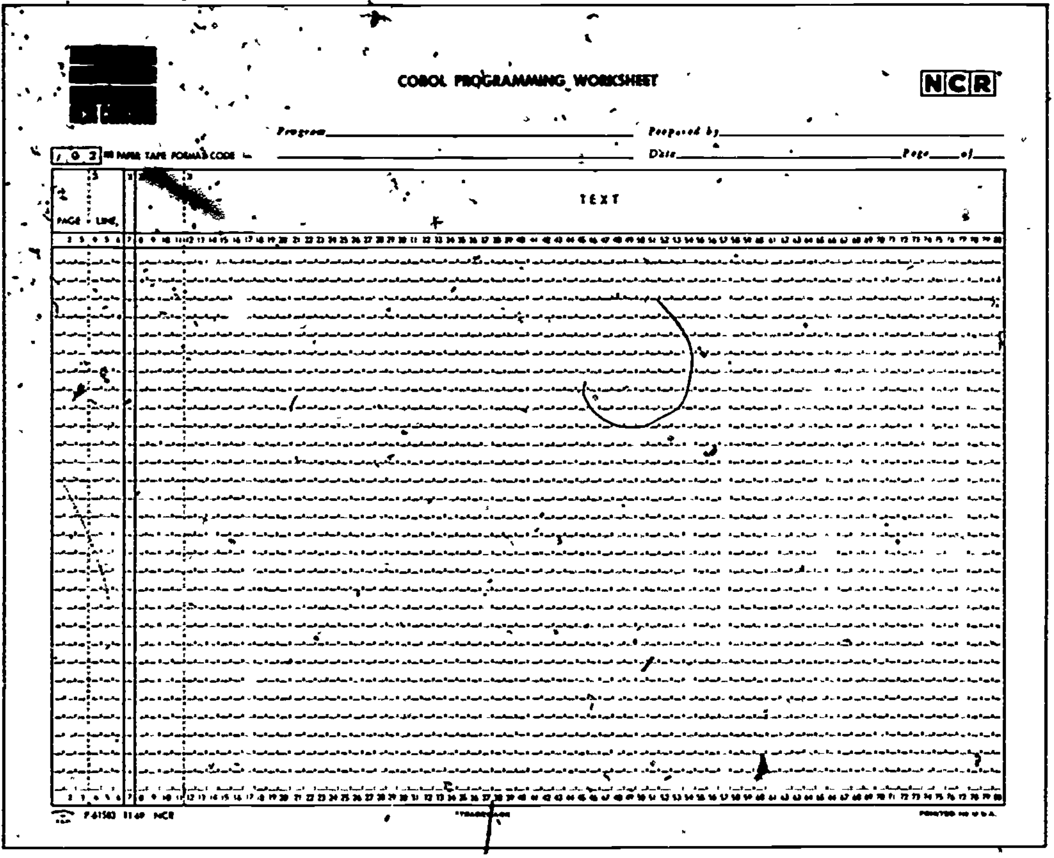
The cards and the sheets may be designed in whatever format is desired. The optical scanner or reader is simply programmed to read from the appropriate places on the form. (These special forms must be professionally printed, so several weeks must usually be allowed.)

When data is first keypunched onto cards or punched onto paper tape, the input of data is more indirect. The forms the data are entered on may range from basic bookkeeping or report type forms to highly specialized forms designed for one user and one system. In either case, the convenience of both the user and the keypunch or paper tape punch operator is an important consideration. It is possible, for example, for a keypunch operator to punch cards from such conventional forms as a school enrollment form or a driver's license application form. This is only true, however, if the operator has a format-key indicating precisely where on the card to punch precisely what data from the form. If many such unspecialized forms are used, it is relatively inconvenient for the keypunch operator.

Therefore, it is far more common for the user of a data processing system to enter data on forms designed with the user and machine operator both in mind. There are some forms available for certain types of information, such as the form programmers use to write their programs on and the forms students might use to write out their class schedule. An example of a form that a programmer might use is shown in Fig. 6-11.

As you can see, there are little numbers at the top and bottom of the programming form. These are column numbers which tell the keypunch operator where on the card to punch the data. A tape punch operator could also punch a paper tape from this form quite easily by following the clearly marked-off spacing of data on the form.

Although such forms are standardized, they are still highly specialized and have very limited uses. But, you can see that such forms



[redacted]

COBOL PROGRAMMING WORKSHEET

NCR

Program ______ Prepared by ______

Date ______ Page ___ of ___

PAGE | LINE | TEXT

F-61503 11-69 NCR

PRINTED IN U.S.A.

Fig. 6–11 Standardized programming form

would be fairly easy to design. For this reason, the systems analyst often designs most or all of the input forms to be used with this system.

In designing input forms, analysts usually follow these guidelines:

1. Minimize the amount of data recording that must be done by
   a. consolidating similar forms.
   b. preprinting as much information as possible on the forms.
   c. using multiple-choice entries rather than open-ended questions. (For example, on a medical form, ask patients to check on a list the diseases they have had rather than to "list the diseases" they have had. This jogs the memory and is easier to code.)
   d. not requesting more information than is needed.
2. Lay out the parts (or fields) of the form in a logical manner. For example,

a. design the form to be filled out from left to right and from top to bottom, the way people are used to reading.
b. arrange the sequence of entries to follow the natural work flow. (Income tax forms, for example, request information in the order in which you will calculate it.)
c. use familiar sequences (address, for example, is usually number, street, city, state, zip).

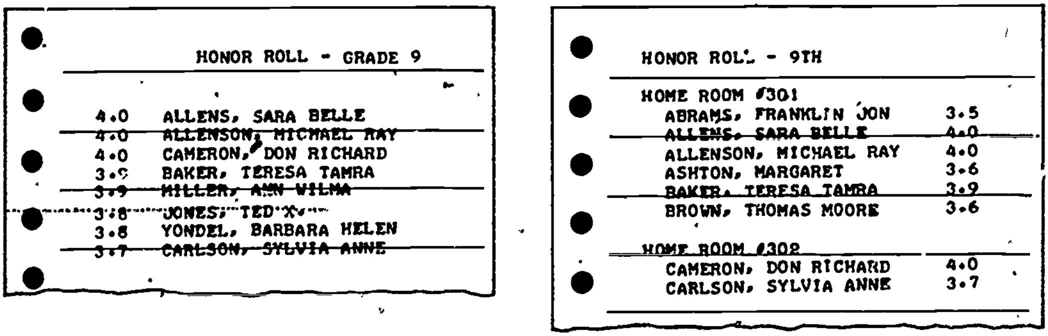
3. Make all instructions clear and precise.
4: When selecting size, shape, and material, keep in mind the way in which the documents will be filed, handled, and transported.

### Designing the output forms

The form in which a data processing system outputs its results is of central concern to both the user and the analyst. The user wants the various pieces of output in a readable and usable form. It is the analyst's job to see that the user gets them in the *most* readable, usable, and economical form possible.

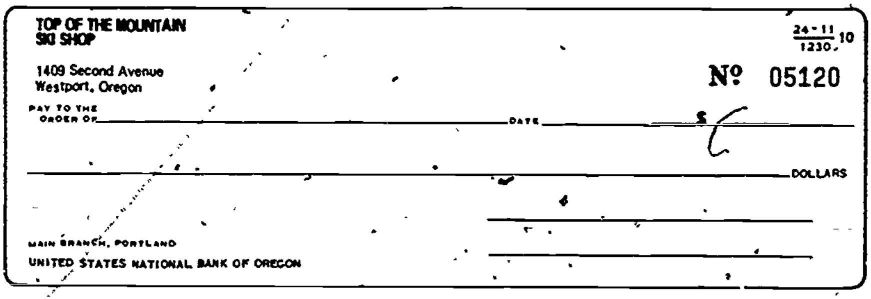
Most systems make use of the computer's regular printer and conventional computer output paper for a large part (or all) of their output. This paper is available on various size rolls which fit into the printer. Whenever the conventional computer paper is used, it is up to the analyst to design the form in which the output will be printed. Here are two examples of the same information being output on conventional computer paper in two different forms:

**Fig. 6–12** Examples of output forms



```
            HONOR ROLL - GRADE 9

    4.0   ALLENS, SARA BELLE
    4.0   ALLENSON, MICHAEL RAY
    4.0   CAMERON, DON RICHARD
    3.9   BAKER, TERESA TAMRA
    3.9   MILLER, ANN WILMA
    3.8   JONES, TED X.
    3.8   YONDEL, BARBARA HELEN
    3.7   CARLSON, SYLVIA ANNE
```

```
HONOR ROLL - 9TH

HOME ROOM #301
  ABRAMS, FRANKLIN JON          3.5
  ALLENS, SARA BELLE            4.0
  ALLENSON, MICHAEL RAY         4.0
  ASHTON, MARGARET              3.6
  BAKER, TERESA TAMRA           3.9
  BROWN, THOMAS MOORE           3.6

HOME ROOM #302
  CAMERON, DON RICHARD          4.0
  CARLSON, SYLVIA ANNE          3.7
```



TOP OF THE MOUNTAIN
SKI SHOP

1409 Second Avenue
Westport, Oregon

$\frac{24-11}{1230}$ 10

№ 05120

PAY TO THE ORDER OF ______________________ DATE __________ $ __________

______________________________________________ DOLLARS

MAIN BRANCH, PORTLAND
UNITED STATES NATIONAL BANK OF OREGON

**Fig. 6–13** Standardized output form for one company

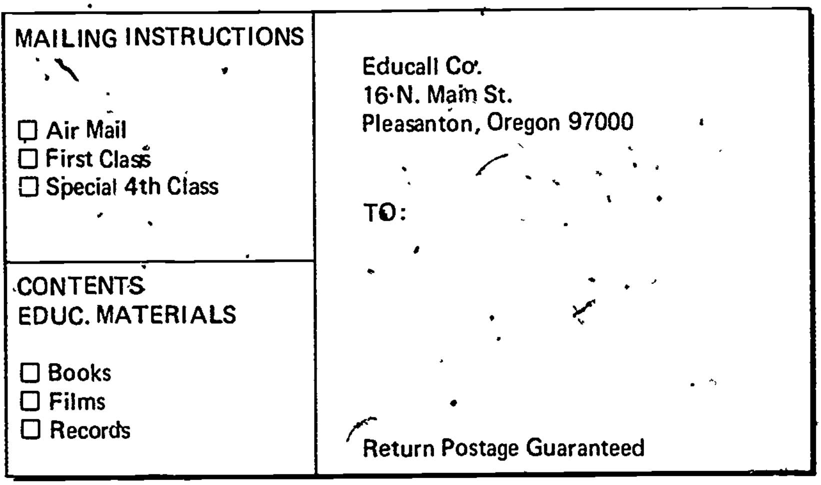
The printer is usually very versatile and can be adjusted to print on almost any reasonable size and weight of paper. For example, it can easily print out onto cards, address labels, and small and large forms.

As in the case with input forms, there are many standardized forms available for output documents such as report cards, paychecks, bills, and so forth.

Whenever they fit the specific needs of the client, the analyst may use such preprinted forms. In many cases analysts design variations of the standard forms or entirely new forms to tailor-fit the user's very special needs. In deciding on the final layout of an output form, analysts usually consider these main points:

1. Is the sequence of the form logical and easy to follow?
2. Are the filing data and margins consistent with the filing equipment or binders to be used?
3. Are the captions on the form easily understood?
4. Is the level of detail appropriate (should items be spelled out, abbreviated, or coded)?
5. Have the space requirements of each field (or section) of the form been verified? For example, is the room allowed for entry of a name or number large enough?
6. Will titles, numbers, or colors make routing, dispatching, or handling easier?
7. Is the size and shape appropriate for easy handling and filing?
8. Is the number of copies correct?

If a user, for example, needed address labels to be output along with other order and sales information, the analyst would want to gather all the facts about how the labels would be used. The analyst



MAILING INSTRUCTIONS

□ Air Mail
□ First Class
□ Special 4th Class

CONTENTS
EDUC. MATERIALS

□ Books
□ Films
□ Records

Educall Co.
16 N. Main St.
Pleasanton, Oregon 97000

TO:

Return Postage Guaranteed

(Actual Size)

**Fig. 6–14** Sample address label

would want to know what packages or boxes the labels would be attached to, what means of mailing would be used, how many copies of the label would be useful (e.g., if one copy could be useful as a packing list), if return postage would be guaranteed, what the regular contents of the packages or boxes are, and so forth. All of this information would be reflected on the final form. Can you see where each piece of data mentioned above has been used in designing the form for address labels as shown in Fig. 6-14?

Careful attention to the details in designing and deciding on input and output forms is time well spent. It can save both the user and the computer center much trouble later on.

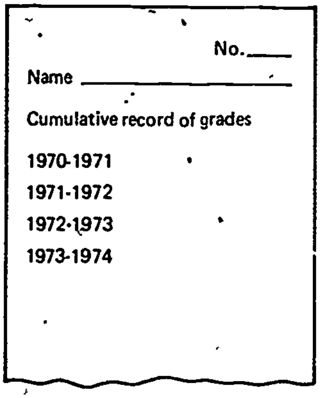
Once suitable forms have been selected or designed, the systems analyst usually orders a limited stock of each at first. Until a system has been thoroughly tested, problems may still appear. Modifications or corrections in the forms may be needed.

## Check your understanding

1. If a keypunch machine is used for preparing input cards,
   a. all input data forms should be designed only for the keypunch operator's use.

b. input data can best be submitted on a standardized mark-sense card.
c. input data forms should be designed for easy use by user and the machine operator alike.
d. none of the above.

2. Assume a client has been using these three handwritten forms:

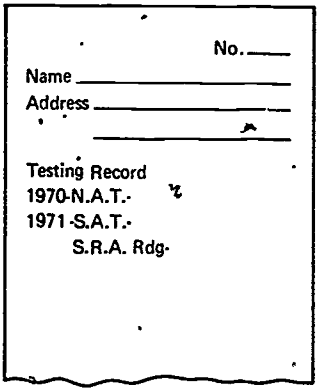


No.____

Name ____________

Cumulative record of grades

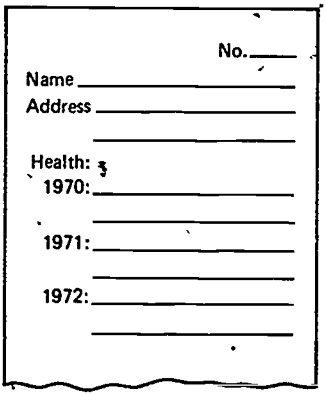
1970-1971
1971-1972
1972-1973
1973-1974



No.____

Name ____________
Address ____________
____________

Testing Record
1970-N.A.T.-
1971-S.A.T.-
S.R.A. Rdg-



No.____

Name ____________
Address ____________
____________

Health:
1970: ____________
____________
1971: ____________
____________
1972: ____________
____________

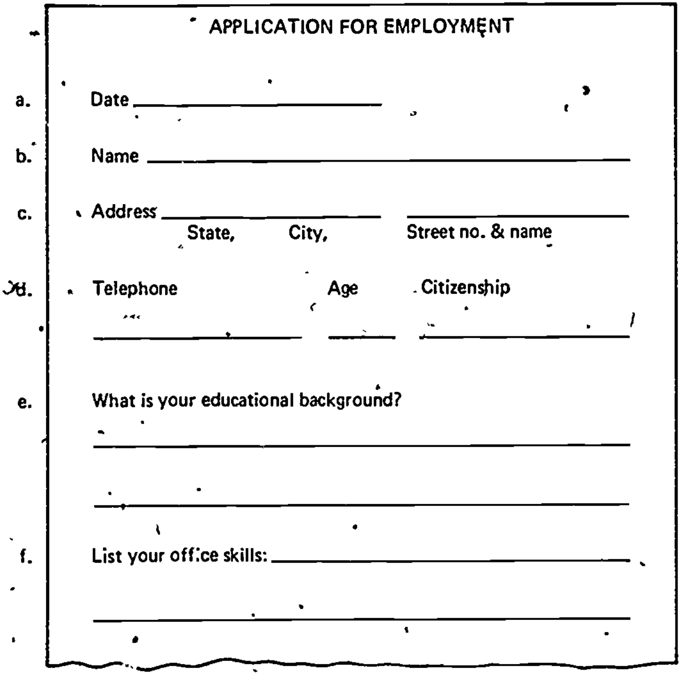
The systems analyst developing a data processing system for the client would most likely

a. use the three forms as they are.
b. design one form for use, consolidating all the information.
c. design three much smaller forms for the same information.
d. redesign the forms so they all have entry places for addresses.

3. A well-designed input form would probably

a. have room for all possible related information to be entered on it, even data not presently needed by the system.
b. be as brief as possible, even if a separate key were needed to decode abbreviations and to give instructions for using the form.
c. be designed in a standard 8 inch x 11 inch size because people are used to using forms of that size.
d. include clear instructions and use familiar sequences for information entries.

4. Identify at least three parts of the input form at the top of page 289 that a good systems analyst would probably change:



APPLICATION FOR EMPLOYMENT

a. Date ____________________

b. Name ________________________________________

c. Address ____________________ ____________________
State, City, Street no. & name

d. Telephone Age Citizenship
______________ ______ ______________

e. What is your educational background?
________________________________________
________________________________________

f. List your office skills: ________________________________________
________________________________________

5. Output forms

a. rarely need to be specially designed.
b. are primarily the user's responsibility.
c. are usually a standard size.
d. are often designed by analysts as variations of standard forms.

6. The regular computer printer is

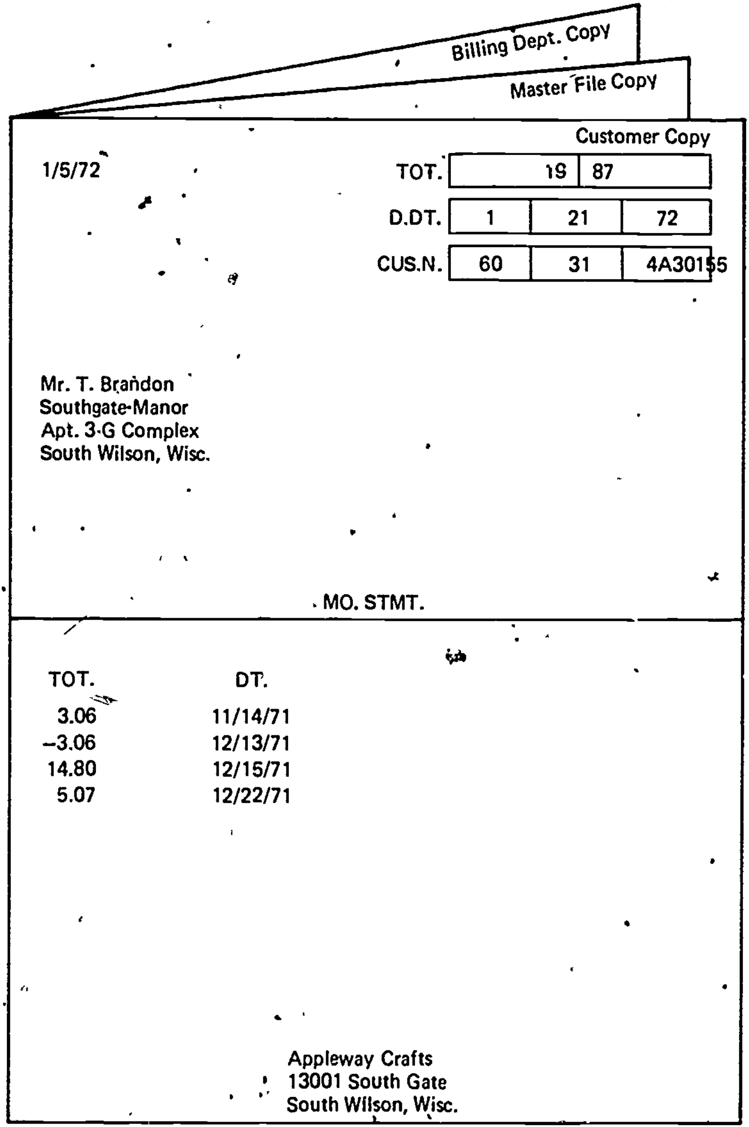
a. highly versatile.
b. able to print on one size paper only.
c. the only output device the computer has.
d. designed especially to fit the user's needs.

7. Which of the following are criteria the analyst is likely to use in deciding on an output form?

a. Size and shape must be appropriate.
b. Sequence of the form should be easy to follow.
c. The form must be adaptable to the user's filing equipment.
d. All of the above.

8. Which of the following criteria have clearly been ignored in designing this output form for a bill?

   a. The space requirement for each field must be verified.
   b. The level of detail should be appropriate.
   c. The captions should be easy to understand.
   d. Is the number of copies correct?



Billing Dept. Copy

Master File Copy

Customer Copy

1/5/72

|  |  |  |  |
|---|---|---|---|
| TOT. | 19 | 87 |  |
| D.DT. | 1 | 21 | 72 |
| CUS.N. | 60 | 31 | 4A30155 |

Mr. T. Brandon
Southgate-Manor
Apt. 3-G Complex
South Wilson, Wisc.

MO. STMT.

| TOT. | DT. |
|---|---|
| 3.06 | 11/14/71 |
| –3.06 | 12/13/71 |
| 14.80 | 12/15/71 |
| 5.07 | 12/22/71 |

Appleway Crafts
13001 South Gate
South Wilson, Wisc.

## Testing the system

The fourth step in the systems approach is that of testing the new system. This usually involves the testing of two main things: 1) whether the system operates correctly and 2) whether the various parts of the system are as useful, economical, and worthwhile as they were intended to be.

Testing a system for correct operation is usually quite straightforward, and problems are normally easy to detect. Since a computer system is run mainly by programs, it is always essential that these programs be tested and debugged before the system is put into full operation. Ordinarily, the programmers will see to this, but the analyst will usually supervise the testing of the entire set of programs when all the pieces of the system are put together. While the set of programs is being tested, problems may show up on the input or output or in the overall form of the system. If they are caught here, they can be corrected before the system is ready for use.

Once the system operates correctly, the analyst can take a careful survey of the overall flow of the system and the output generated to see that all aspects of the system meet the criteria of both usefulness and economy. During this survey the analyst may discover that a desired output is too difficult or expensive to produce to make it worthwhile. The limitation might be the cost of running a certain program, time or cost involved in coding, a difficulty in running or distributing a certain report, or effort required to obtain the input data. (Let us demonstrate this by an example. You might like to have a weekly, or even daily, report of your checking account balance mailed to you by your bank. The information exists, and it would be a simple programming problem to produce such reports. But, obviously, the cost of computer time, paper, mailing, and so forth would be so high that the report would not be worth the cost.)

Compromises must be made. It is a common fault of some analysts and programmers to want to create highly sophisticated systems that can do everything, just because it is possible to do so. One of the major tasks of the systems analyst is to balance this enthusiasm with the realities of budgets.

Testing actually continues once the user begins to operate the new system. During the system's initial use, it may be discovered that some of the required input is not as readily available as the client thought or that some of the output documents are not needed after all. These discoveries may call for changes in the system.

Because the testing of a system normally does continue on into the client's initial use of it, analysts commonly advise the user to start out by running the new system along with the old one (whether it was automated or not).

This ensures that if there are problems with the new system, the work will still be done correctly. It also allows comparison of data as a check on the accuracy of the new system. Running parallel until everyone is satisfied that a system will function smoothly relieves much anxiety about a change in systems. It is usually the last step before the client accepts the job as completed and implements the system fully.

## Putting the system into effect

***"Selling" the system.*** No, we don't mean systems analysts must also be computer sales-people! But, a very important step in putting a system into effect is to gain the support and cooperation of the people who must operate the system once it is installed. Without that support and cooperation, the system is sure to fail.

A good example of the ingenious ways people will sabotage a system they don't support is in the use of seat belts in the newer cars. Unless one seat belt is fastened for every front seat passenger, the car won't start. Those people who don't agree with the need for wearing seat belts for auto safety will go to any lengths to see that the system won't work properly. They will sit on the forward edge of the seat to avoid triggering the pressure switch that senses the presence of a person in the seat. They will pull the belt out of the retractor and tie a knot in it or hook it over the door lock to keep it fully extended. They will fasten the seat belt and then sit on it to keep it extended. Or they will simply pay a garage $15 to disconnect the whole system!

It is this well-known aspect of human nature that makes it so important for a systems analyst to spend time planning and working with the people who will operate the system, to win their confidence. Operations personnel who have the major responsibility for making the system a success or a failure must understand and accept the need for it, be in on its development, know how to operate it, and feel that they are an important part of the system. It is up to the systems analyst to enlist the support of operations personnel early in the project.

Not only operations people, but anyone who will be involved in providing or preparing data, receiving and interpreting output, or who will be in some way affected by the change should be involved in an orientation program. Such a program would explain the need for the system, the benefits to be gained, and the new methods to be used.

For example, consider a school where teachers fill out report cards by hand for each student. If that school were to install a computerized grade-reporting system, who would need "selling"? First, the students. Then the teachers, who would submit grades in a new way which at first seems more complicated than the old. Parents would have to adjust to reading a computer-produced grade report rather than the more personalized hand-written report. Others would also be affected by the new system—school secretaries, counselors, and administrators. Failing to explain and justify the system to these people could cause many problems for the systems analyst. It is not enough that a system is technically workable. It must also be understood, accepted, and supported by the people who use and operate the system.

*Documentation.* Before clients can use new systems, they must have adequate materials to explain in detail their part in making the systems run smoothly. These are prepared by the analysts, as the "documentation" of the systems.

In general, the analyst is expected to provide materials such as the following:

1. A thorough general description of the system (called a procedure-manual) for the client. This includes notes on the overall paper-flow and the output. Where appropriate, guidelines would be given for obtaining the forms used with the system.
2. Complete listings and sample runs of all programs. (Incidentally, the computer center for which the analyst is working normally also keeps a copy of all the programs for each system developed.)
3. Manuals on how to use all the input forms, including samples.
4. Manuals on using the system's outputs, including samples.
5. Handbooks with instructions on data preparation and specific operations. The first would include such information as how to keypunch from specific input forms. The second would specify such things as approximate running times for each program, when to load which output forms, what card decks or magnetic tapes to load and when.

In addition, if a system is modified or updated at a later time, it is the responsibility of the systems analyst to see that all documentation is similarly modified or updated.

*Implementation.* The implementation phase is the one in which the system is set up in the user's installation and the systems analyst sees to it that the system functions smoothly. If a system has been thoroughly tested, this should happen rather rapidly.

The analyst may be involved in training of personnel to work with the new system. While this often begins earlier in the design and testing phases, it frequently continues into the implementation phase.

Once the system is installed and running smoothly, the analyst may have little further involvement with it. More frequently, however, a systems analyst finds that problems develop and have to be solved, or the system must be constantly reevaluated and revised to keep up with the changing needs of the organization.

## Check your understanding

1. Once the programs for a system have been tested by the programmers,

   a. the analyst usually supervises a final testing of the entire set of programs.
   b. the programs are considered finished.
   c. the programmers conduct a final testing of all the programs together.
   d. none of the above.

2. Once the user begins to operate the system,

   a. the system is considered fully implemented.
   b. the analyst begins to prepare documentation for the system.
   c. any previous system used is dispensed with.
   d. testing of the new system continues for awhile.

3. Documentation of a system

   a. is usually done by the user after he or she starts operating the system.
   b. is usually done by the analyst before he or she starts testing the system.
   c. is always the last step in a system analysis.
   d. none of the above.

4. Which of the following is not a part of a systems documentation?
   a. Manuals on using input forms
   b. Handbooks on operations like setting up output devices and preparing input media such as punched cards
   c. Guidelines on how to program for the system
   d. General description of the system and its parts

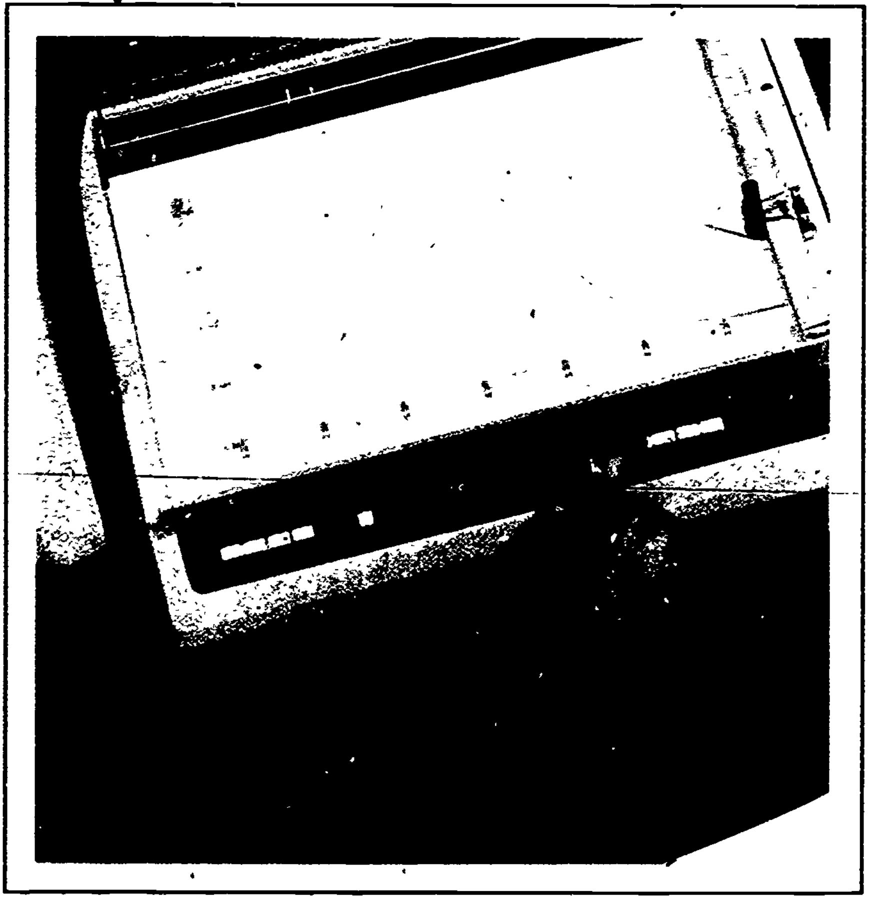
5. The implementation of a system often involves
   a. training personnel to use the system.
   b. continued testing of the overall system.
   c. the operation of the new system along with the old system for a time.
   d. all of the above.

6. Which of the steps listed below, *if left out*, would be most likely to cause the system to fail?
   a. Designing decision tables
   b. Selling the system
   c. Providing sample runs
   d. Providing for all desired output

7. To summarize your understanding of the systems approach, write the following steps in the correct "systems approach" order.

   Design the system
   Test the system
   Define output
   Document and implement the system
   Define the elements of input

# 1234567



# Computer careers in perspective

## INTRODUCTION

Perhaps you would like to explore a career in a computer-related area. What do you need to know to begin your exploration?

First, you need to know what jobs are available. Then you should explore the nature of the job, the kind of work involved, the working conditions you might expect, the requirements for the job, and the prospects for future employment and advancement. Finally, you need to carefully take stock of yourself. Will your particular interests, capabilities, and personality "fit" the job you have in mind?

What type of person are you? Don't just ask yourself. Ask your friends, teachers, parents, and employers for their viewpoint. Their different perspectives might provide you with valuable insights about yourself. Do you enjoy detail work? Like to solve puzzles? Need or want close supervision? Enjoy working with people? Like to "run things"? Are you interested in machinery? Ambitious? Well-organized? Your answers to all of these questions will give you some clues about the kind of computer-related career you might want to explore further. You may even wish to begin preparations for a future career in the computer field.

In this chapter we will discuss several computer-related careers: data preparation clerk (or data clerk), computer operator, computer programmer, systems programmer, systems analyst, and computer center manager. Chapters Two through Six dealt with the kind of detailed knowledge and skills you would need in order to pursue any of these careers. In this chapter, each of the careers is examined separately to indicate such things as what the job is like, what its requirements are, and what it involves in the way of working conditions and prospects for the future. All of these careers are placed in perspective to help you make decisions about further exploration and preparation.

## SALARY LEVELS

It would be a waste of time to state the salaries that could be expected in any of the careers discussed. Salary levels change every year, so any figures given here would be outdated before the book was even published! For example, the average salary of a computer programmer in 1960 was around $110 a week, in 1965 around $130 a week, and in 1970, it was around $180 a week. In 1976 . . . ? Similarly, a keypunch operator who earned about $65 a week in 1960, averaged $85 a week in 1965, and in 1970 averaged $100 a week.*

It might be more informative to compare salaries of computer personnel with the salaries of people in other types of careers. Figure 7-1

* Current salary levels for all computer-related careers are available from the U.S. Department of Labor, Bureau of Labor Statistics, Washington, D.C.

shows various careers that are comparable, in terms of expected salaries, to some of the computer-related careers we are exploring.

In management (such as of a computer center, computer operations, or systems and programming) salaries will vary so widely that an inexperienced manager may make less than an experienced computer

**Fig. 7-1** Comparison of salary levels

| COMPUTER CAREER | CAREERS WITH COMPARABLE SALARY LEVELS | |
|---|---|---|
| Data Preparation Clerk | Licensed Practical Nurse<br>Medical Technician<br>Service Station Attendant<br>Office Machine Operator<br>Telephone Operator<br>Bank Clerk, Teller | Cashier<br>Taxi Driver<br>Stenographer<br>Custodian<br>Waiter or Waitress |
| Computer Operator | Photographic Darkroom Tech.<br>Elementary School Teacher<br>Furniture Upholsterer<br>Commercial Artist<br>Executive Secretary<br>Telephone Installer<br>Merchant Seaperson<br>Registered Nurse | Dental Hygienist<br>Welder<br>Chef<br>Barber<br>Mail Carrier<br>School Counselor<br>City Busdriver<br>Diesel Mechanic |
| Computer Programmer | Occupational Therapist<br>Newspaper Press Operator<br>Secondary School Teacher<br>Automobile Salesperson<br>Computer Service Tech.<br>Radio or TV Announcer<br>Construction Laborer | Jeweler, Watchmaker<br>Social Worker<br>Musician<br>Flight Attendant<br>Truck Driver (Local)<br>Purchasing Agent<br>Police Officer |
| Systems Programmer | Food and Drug Inspector<br>Air Traffic Controller<br>Anthropologist<br>Locomotive Engineer<br>Newspaper Reporter<br>Photographer<br>Sociologist<br>Engineer | Surveyor<br>College Professor<br>FBI Agent<br>Accountant<br>Credit Official<br>Lawyer<br>Personnel Worker |
| Systems Analyst | Real Estate Salesperson<br>Long-distance Truck Driver<br>Forest Ranger | Plumber, Electrician<br>Bricklayer<br>Carpenter |

operator. At the same time, a high level manager, with experience and much responsibility (for example, a manager in software development) could make as much as a chief hospital administrator, a veterinarian, a chiropractor, a pilot, a dentist, a city manager in a large city, or a chief of police.

Now, let's go on to take a closer look at the world of the data clerk, computer operator, systems programmer, systems analyst, and center manager.

## THE WORLD OF DATA PREPARATION

### Introduction

In the world of information (or "data") processing, the data preparation clerk, or data clerk, holds an important and responsible position. To get a general idea of where this position fits in the overall picture of data processing, look at the diagram in Fig. 7-2.

As you can see from this diagram, the first stage in processing data involves the computer programmer, who writes the program (instructions for the computer), and the customer, who supplies the data to be processed. Once the programs and data (in handwritten or typed form) are gathered and ready for processing, they are turned over to the data clerk who "prepares" them for loading into the computer by transcribing them onto a medium such as punched cards. The data (including programs) then goes to the computer operator, who runs the job. If there are no errors in the data, the results of the processing (in this case, a report) can be delivered to the customer.

In general, the data clerk is responsible for transcribing all data (including programs) onto a computer-compatible medium such as punched cards or tapes that can be fed into the computer.

Information can be fed (or "input") into a computer through several different kinds of input devices such as card readers and paper tape readers. It is the data clerk's job to transfer the raw data (programs and data that are in regular handwritten or typed form) onto the cards or paper tape that can be read by these devices.

Data clerks most often work with machine-marked media such as punched cards and paper tape. The machines used to prepare these media are usually run from keyboards on which the clerk types the data.

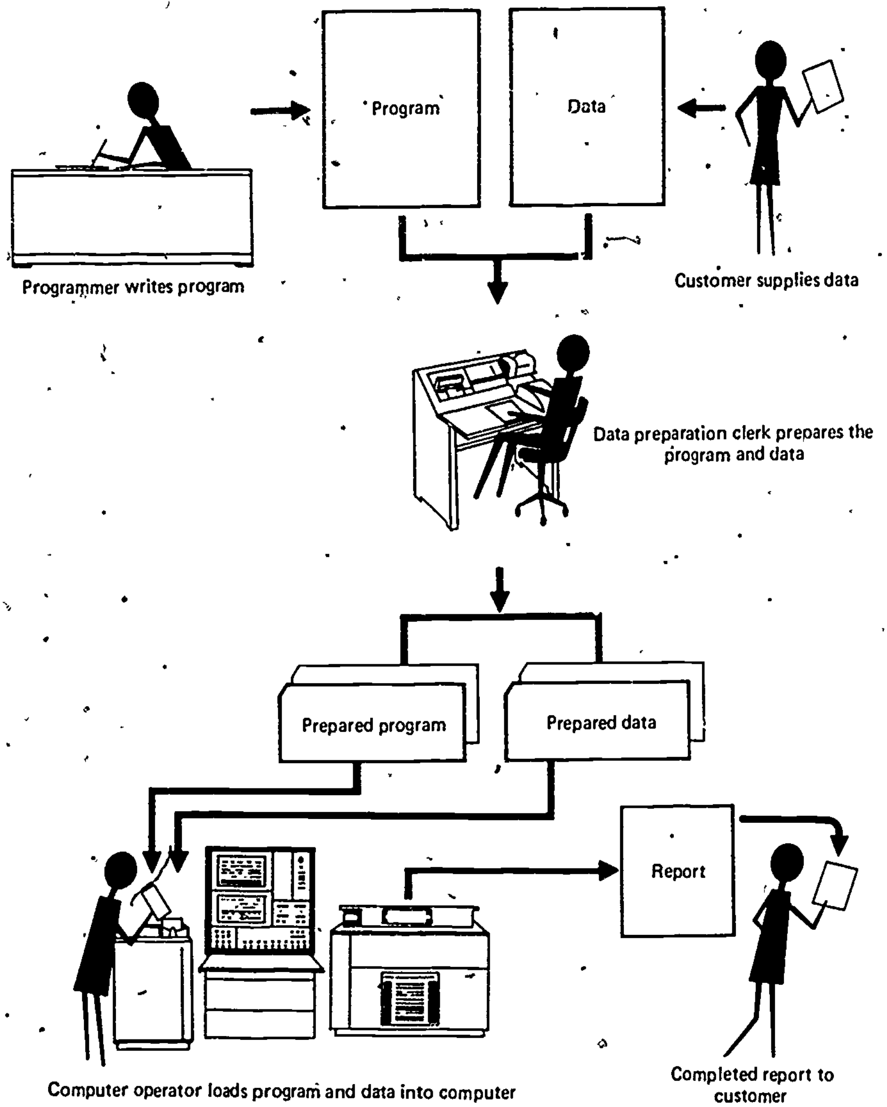

Programmer writes program
Program
Data
Customer supplies data
Data preparation clerk prepares the program and data
Prepared program
Prepared data
Report
Computer operator loads program and data into computer
Completed report to customer

**Fig. 7–2** The flow of data processing

The data clerk's work is always guided by careful and complete instructions provided by the programmer. When the programmer turns the raw data over to the data preparation department, specific instructions concerning the medium to be used and any other directions the data clerk needs to transcribe the data correctly are always included.

After the data have been transcribed, the cards or tapes are "verified" (that is, checked for correctness) either by the same clerk or by another clerk. At this point, all errors discovered in the data are corrected. Then the cards or tapes are routed to the computer operator for processing.

When the job is run, errors in the prepared data can cause problems in the processing. For instance, the computer may stop running and/or errors may occur in the output (or results) being produced. In such cases, the cards or tapes are returned to the programmer. The programmer must locate the errors causing the trouble. Whether the mistakes originated in the raw data or in the preparation process, the programmer indicates the corrections needed and has the data clerk transcribe and verify new cards or tapes. Once corrected data is transcribed, the processing job can be done by the computer operator.

You can see that the work of the data clerk is a vital link in the chain of data processing activities. In fact, if raw data cannot be prepared for processing accurately and quickly (by skilled data clerks), a computer center will find its entire production held up. An efficient computer center always owes some of its speed and economy to the skill of its data clerks.

## Requirements of the job

*Skills.* Data clerks must be skilled at typing, since most of the data preparation machines are run from typewriter-like keyboards. In addition, he or she needs to know in general how to operate the various data preparation machines, including how to load, run, and unload them. To qualify as a skilled data clerk, both speed and accuracy in using the data preparation machines are required.

Forms and different kinds of media, such as cards and tapes, are common in the work done by the data clerks, so the ability to use and handle them efficiently is a must for the competent data clerk.

Although there is generally a data preparation department supervisor in charge of assigning jobs and scheduling work, the individual clerks should be skillful in organizing their time to meet the overall schedules.

*Education.* The better data clerks usually are high school graduates. Although this may not be required on some jobs, it is desirable in order to provide reasonable assurance of promotion. If a person is a good typist, the formal training in the use of the equipment will take only about one week. Three to four weeks will be needed on the job to develop production speed and accuracy. The person who does not know how to type must learn typing before beginning the formal training. This will require four to six weeks of intensive training on typing skills before training for the career of data preparation is begun.

*Personal characteristics.* A look at the professional data clerk will reveal a responsible, alert, and practical person. He or she can understand and follow specific instructions easily and accurately and perform repetitive work without becoming bored. Manual dexterity coupled with a desire for accuracy and perseverance mark the data clerk as a true professional as well as an important member of the computer group.

*Working conditions.* The data preparation installation is clean, orderly, and well-lighted. Being a part of a busy production unit, data clerks usually have little time for conversation. Outside distractions are discouraged as they may decrease the efficiency and accuracy of the work. Some large installations operate day and night, thus needing several different crews of people working shifts. New employees often are required to start on night shifts and "work up" to day work.

The machines used in data preparation are usually arranged in rows for greatest efficiency in the flow of the job. Because of the nature of the machines, the noise level in any installation is likely to be fairly high. Some machines, like sorters and reproducers, generate more noise than others. Whatever specific machines a given data preparation department may have, the noise level is usually higher than that found in most general offices.

Data clerks are responsible to and receive directions from the data preparation supervisor. The assignment and scheduling of all work come from the supervisor, who receives data to be prepared from programmers, systems analysts, or even outside sources. The supervisor generally oversees the work of the data clerks and is usually responsible, in turn, to the Operations Manager or the Computer Center Manager.

## Data preparation as a career

The job of data clerk is essential to the operation of any computer center. Raw data must be prepared for entry into the computer, and prepared data usually needs verification and often requires sorting. In some cases, prepared data also needs to be reproduced. Often, raw data must be arranged and organized before it can be prepared on cards or tapes. All of these essential tasks continue to require skilled data clerks.

Most computer centers have several data clerks in their data preparation department. Large centers may employ up to ten or more data clerks to keep up with the data preparation work load. In general, as computer installations grow in number and volume of work handled, job opportunities for skilled data clerks will be increasing.

The job of data clerk, being an essential job in a growing field, is one which provides good security for the future. Many professional data clerks continue on in the clerical position. They increase their skills and expertise on the job and frequently learn to operate more sophisticated data preparation devices. From the position of data clerk, the worker can also move to other clerical positions in the computer center or into secretarial jobs. For the promotion-bound clerk, superior skills and competence in the clerical position can be a strong recommendation for promotion to the position of data preparation supervisor or computer operator.

## THE WORLD OF THE COMPUTER OPERATOR

### Introduction

In the computer center, the person who works most directly with the computer and its related equipment is the computer operator. In fact, she or he is responsible for the efficient use of the computer itself and of the additional devices which feed information into and out of the computer. No computer center can function very efficiently without a competent and skillful computer operator.

When the computer center's programmei has finished writing the program for a specific task (or "job," as it is called) and has gathered together all the data to be used with the program, the programmer is nearly ready to put the "running of the job" into the computer operator's hands. Of course before this, the program and the data must be transcribed onto a computer-compatible medium such as punched cards or paper tape. This is usually done by a data clerk.

Once ready to be loaded into the computer, these paper tapes or decks of punched cards are taken to the computer operator along with an "instruction sheet" giving any special instructions involved in running the program. Here the computer operator's work starts.

The computer operator first schedules the running of the job on the basis of the job's importance and the availability of the computer's time. Then, at the scheduled time, the operator sets up the computer equipment to run the job as directed by the instruction sheet. This includes picking out and mounting appropriate printout paper on the computer printer and loading the input media (cards, paper tapes, etc.) onto the appropriate input devices (card reader, paper tape reader, etc.). When all the card or tape input for a job is attached to the instruction sheet by the programmer, the computer operator has it automatically at hand when ready to run the job. In many cases, however, the programs and some of the data needed for a job are stored on cards or tapes kept in the computer room. In these cases, the operator must gather the required input from shelves or files and then load them for use at the appropriate time.

Once the equipment is correctly set up for the job, the operator loads the program into the computer or calls it up from computer storage, so that the computer is prepared to do all the proper things to the data that is to be processed. Then, the operator starts the computer "run" and monitors the computer's operations from the control panel and the console typewriter. As you already know, the console typewriter allows the computer operator to type in directions to the computer as required by the instruction sheet, such as telling it initially to start the run. The console typewriter is also used to type in information which the program may request, such as the current date. The control panel is equipped with buttons, switches, and lights which inform the operator what the computer is doing and which allow regulation of its operation.

When the control panel lights show that the machine has stopped for some reason, the operator has to determine if there is a problem. If there is a problem, the operator must correct it or call in a maintenance engineer. He or she also monitors the control panel, the console typewriter, and the output (data printed out by the computer) for any evidence that the run is not proceeding correctly, stopping the job if necessary.

Whenever a run cannot be finished because of a problem in the program or data, the operator returns the cards or tape to the programmer. The instruction sheet and a brief but precise report of all that was observed that might indicate where the problem may lie is also returned to the programmer.

When a job has been successfully run, on the other hand, the operator removes the cards or tapes used for the job and stores them in the proper place in the computer room. The instruction sheet and all output is given to the programmer. If the programmer has requested that cards or tapes also be returned, they are returned with the instruction sheet and output.

As you can probably imagine, computer rooms are usually very busy places and time is valuable. For this reason it is important for the computer operator not only to schedule the work carefully, but to keep an accurate log of the time used for each job. The computer operator is also responsible for writing periodic reports on the use of the computer she or he is in charge of.

In large computer centers, the computer operator usually has some help in keeping the computer room running smoothly and efficiently. Most often one or more tape librarians will help with the location and handling of the magnetic tapes. There may also be assistants trained to operate some of the other equipment, including the input devices which feed information into the computer and the output devices, through which information comes out of the computer. These input and output devices include such equipment as card readers, high speed printers, and separate card handling equipment such as sorters and collators. In most small centers, however, the computer operator runs the computer and all the peripheral equipment alone, and does all the handling and filing of tapes, disks, and cards stored in the computer room.

## Requirements for the job

*Skills.* Becoming a competent computer operator requires the development of several special skills as well as the development of some more general ones.

Of the special skills needed, the most important are those involved in setting up the computer equipment to run any given job and in operating the computer from the control panel and console typewriter. Both of these tasks require a basic knowledge of the various pieces of computer equipment and how they work together. Naturally, the better the computer operator understands the equipment and the way each piece works with the others, the more efficiently he or she can run the computer system and the more quickly any trouble that might occur can be located.

In addition to operating equipment in the computer system, it is helpful if the computer operator knows how to run the various machines which prepare and handle punched cards, including keypunch machines, card sorters, card interpreters and verifiers, and card collators. The computer operator must also be proficient in the general handling and efficient storing of card decks, tape reels, magnetic disk packs, and other input/output media commonly kept in the computer room.

Good skills in basic arithmetic are important to computer operators. Since computer printouts commonly involve numbers and results of simple arithmetic operations, operators sometimes check the printout during the run for evidence of any errors or problems. It is easy to see that computer operators who can catch arithmetic errors at a glance can save a considerable amount of valuable time, both for themselves and the computer.

Another skill of considerable importance to the computer operator is that of communicating effectively both in speech and in writing. Both forms of communication are used continually in the work. This is because the computer operator has to understand directions from and write explanations to programmers about specific computer runs. The operator must also communicate with the computer through the console typewriter, and write periodic reports about the computer's use.

*Education.* The minimum education expected for the computer operator's position is a high school diploma. The candidate for the position should have attained a useful general education and a good background in mathematics as well as in language arts. Most computer operators find that the high school courses which helped them the most were mathematics and English courses. Logic and psychology provide additional useful background.

Further, more specialized training for the job is desirable and can be a real boon to the high school graduate interested in a computer operator's career. Such training is readily available in many vocational high schools and colleges, as well as through classes offered by computer manufacturers. Computer centers may also make on-the-job training available.

Like others in the data processing profession, the computer operator's education doesn't stop after getting the job. Continual changes in the data processing field in general and in computer equipment in particular make further training through on-the-job and off-hour courses necessary from time to time.

*Personal characteristics.* Since the operation of machines is the "heart" of the computer operator's work, enough natural mechanical ability to feel comfortable learning about keyboards, control panels, and the like is needed. In addition, a computer operator must be able to use his or her hands easily because handling cards and tapes and operating keyboards and control panels are constant activities. Further, since the computer operator often has to lift and carry card decks, rolls of printout paper, disk packs, and other bulky items, he or she needs to be generally physically fit.

The setting up and monitoring of the computer system requires constant attention to details. For this reason, the computer operator needs to be naturally alert and should enjoy doing detailed and accurate work.

The computer room is a busy, sometimes hectic, place to work. There are usually hour-to-hour (sometimes minute-to-minute) changes in computer use, and work is frequently done under the pressure of deadlines. Sudden "rush" jobs may add to the bustle. Consequently, the computer operator must be adaptable and able to keep a cool head, even under pressure.

Finally, the computer operator should be able to get along well with people. Often the operator has to work with programmers and systems analysts in scheduling and running their jobs, and with assistants in the computer room.

*Working conditions.* The computer room is the site of most of the computer operator's work. It is usually clean, well-lighted, and air-conditioned, with moderate noise level from the equipment. Smoking is generally not permitted in the computer room. The computer operator normally observes the routine eight-hour day with morning and afternoon coffee breaks and lunch hour breaks. However, it is not uncommon for special rush jobs or temporary computer breakdowns to make the operator's work schedule irregular.

Although the computer operator does a certain amount of work at his or her own desk, such as writing summary reports and filling out logs and schedules, most working hours are spent standing and moving about the computer in the process of setting up the equipment and monitoring its operation. In the midst of runs, he or she has short periods of inactivity, but these nevertheless require constant alertness because the equipment must continually be watched to be sure it is running properly.

The operator generally functions as the overseer of the computer equipment, assisting programmers in testing their programs and making final runs of jobs on the computer. He or she usually reports directly to the operation supervisor who, in turn, is responsible to the manager of the computer center. In larger centers where several computer operators may be employed, there is often a "lead operator" who guides the other operators in their work.

## Computer operation as a career

Since more and more organizations are installing computers or using computer services, the demand for computer operators is generally increasing. This is particularly true in the metropolitan areas where the largest businesses and industries are located and where the turnover in operating personnel is relatively high.

In the larger cities, candidates for a computer operator's job may apply to any number of different kinds of organizations for possible employment. Colleges and universities; data processing firms; large corporations or businesses; banks; insurance companies; city, county, and state governmental offices; and the offices of major utilities are only a few possibilities. Applicants may also want to consider taking positions as input/output clerks and decollators, tape librarians, or scheduling clerks, for which they are usually well qualified.

There are several directions a computer operator's career can take. Many successful computer operators choose to continue working directly in computer operations, where changes and improvements in equipment allow them to keep expanding their knowledge and skills. In larger centers, operators may move into the position of lead operators and, if they have supervisory abilities, they may move upward to operations supervisors. Other accomplished computer operators move into programming careers, preparing themselves either through outside programmer training courses or through on-the-job training.

On the whole, the successful computer operator has good chances for promotion. Knowledge of computer equipment and how to operate it effectively provide a good base from which to learn other specialities in the data processing field.

# THE WORLD OF THE COMPUTER PROGRAMMER

## Introduction

When customers come to a computer center, there is usually only one thing they know for certain: that they have problems which a computer can probably solve. The problems might deal, for example, with payroll or accounting, with statistical analysis, or with attendance or grade reports. Whatever the particular needs are, the customers' understanding of their situation is usually quite vague at first. They ordinarily know what they want (for example, report cards or paychecks or a scheduling system), but they may not have a very clear idea of the best form for the computer solution, and they usually have almost no idea at all of how to get a computer to produce what they need.

The first step in applying a computer to the solution of the customer's problem is for the customer to explain what is needed to a systems analyst at the computer center. It is up to the systems analyst to analyze all the aspects of the problem and to define the job involved in solving it. Once the customer's problem has been assessed, the task is assumed by the computer center. While the customer may continue to provide information pertinent to the problem, it has in effect been "turned over" to the computer center staff.

At this point, the systems analyst is responsible for exploring possible approaches to the problem. He or she must choose one of the approaches and then create a master plan for meeting the customer's needs as determined.

When the systems analyst has finished designing the master solution plan, the problem is by no means ready to go to the computer. The master plan is an overall plan showing the general steps to be taken in solving the problem. For the computer to solve the problem, however, it must have very specific input. This includes the exact data to be used in the solution and step-by-step instructions on how to manipulate the data and produce the results. So, from the systems analyst's master plan, we go to a second step—and to a second person, the computer programmer.

*Specifications for the job.* When the systems analyst calls in the computer programmer to begin writing the computer programs for a customer, the analyst provides the programmer with a set of instructions (called "systems specifications") which include information about the following:

- *Input*—The kind of information (or data) the computer needs to perform the task, and the form it may be put in
- *Output*—The information the computer is to produce, including the kind of format it must be produced in
- *Processing Requirements*—Description of any specific selection, or calculation processes (for example, mathematical formulas) required to produce exactly the output needed

With the systems specifications in hand, it is up to the programmer to work out the process by which the computer will use the input to produce the output. In so doing, the programmer creates the step-by-step instructions (called the "programs") for the computer.

*Approaching the programming job.* The first thing the programmer must do is to plan a method for solving the problem. To do this, a technique called flowcharting is used. From the information given, the programmer begins by blocking out the general steps in the solution to the problem. Then he or she refines those steps into clearer, more precise ones. The programmer finally ends with a detailed chart showing the complete flow of steps in the solution to the problem. This is the flowchart of the solution.

Next, the programmer must translate the flowcharted plan, with its logical sequence of steps, into a set of coded instructions for the computer. By "coded" we simply mean that the everyday language the programmer thinks in is translated into a code language or "shorthand," which the computer can then translate into machine language. There are numerous different "shorthand" languages which can be used in programming and, logically enough, they are called "programming languages." You already know a few names of programming languages —for example, FORTRAN and BASIC.

Once the program is on cards or paper tape, the programmer then proceeds to use a computer to see that each program step is correct and can be understood by the computer. There are several steps involved in this procedure which is generally referred to as "debugging" the program.

When a program has had all the "bugs" taken out of it (errors in the preparation of cards or tape, errors in the programming sequence of steps, and so forth), the programmer goes on to test it to see that it produces the needed results correctly. If any problems are encountered in the testing of a program, of course, the programmer will correct the program, debug it again, and retest it until it works perfectly.

Once the program has been debugged and successfully tested, it is considered a finished program. Before the programmer gives the program to the computer operation staff for use, however, all necessary descriptions about what the program does, how it works, and how to run it are written up. This is called "documentation" because it involves the writing of documents to go along with the program.

### Requirements of the job

The job of computer programmer requires various skills, education, and personal characteristics. The programmer is the member of the computer center team who translates the broadly described system for solving a problem into the specific, well-defined tasks necessary to accomplish the job. He or she is also the person who directly communicates with the computer.

*Skills.* The programmer's knowledge of the computer is more intimate than the knowledge of those who operate it, for the programmer must "speak its language." Each computer can understand certain sets of symbols—called a programming language—and can translate those symbols into its own machine language or code. A programmer must be able to write programs in at least one of the programming languages currently used with the computer.

While the computer operator is the expert on running the computer hardware, a programmer needs to know enough about hardware to code a process for it. He or she must know how the hardware components operate as well as what their limitations are. There are times when the programmer may also be required to prepare data for the programs or run programs on the computer.

In testing a program, one must be able to analyze errors or problems, to determine when a program is processing correctly. The "debugging" and testing phases require a patient, methodical approach to the situation. The computer programmer will do well to have a taste for logic and order and an enjoyment of puzzles and problems. When working out a program, a programmer must persevere until it is correct.

He or she may also have to exercise judgment in rejecting certain procedures and working out new ones, often under the pressure of deadlines.

Once a program has been developed, it is the computer programmer's responsibility to document—that is, to write detailed descriptions of the processes involved. These thorough descriptions provide an explanation of the reasoning and logic in the program, and also include information about preparing data, running the job, and verifying (checking accuracy). Although the programmer must have the verbal skills necessary to communicate with the systems analyst and the operation staff, the programmer will seldom deal with those outside the computer staff. It can be assumed that this work will most often relate to others who are familiar with computer operations.

*Education.* Although a beginning computer programmer may have no more than a high school diploma, many organizations prefer some college or technical school background. A college degree is especially desirable for a scientific programmer and those programmers who wish to become experts in the development of software (the internal programs in the computer).

Most programming jobs do not require extensive mathematical training, but they all require the ability to understand fundamental number relationships and arithmetic.

In training for the job, a programmer may learn about several aspects of data processing. Most programmer training courses offer the basic elements of computers and their operation as well as specific training in one or more programming languages.

Community colleges and technical schools offer data processing courses which provide good preparation for the computer programming trainee. Training may range from an eight- to ten-week programming language course to a two-year data processing course. On-the-job experience for six to twelve months as a trainee is usually necessary for the programmer to assume the full responsibilities of the job.

*Personal characteristics.* Computer programmers should be orderly and meticulous in their work habits. A logical nature and the ability to analyze problems enable the programmer to evaluate any job assigned and to break it up into specific tasks. Graphic and verbal skills help her or him to efficiently design detailed flowcharts and write the documentation for any assigned job.

Programmers should be self-motivated. They must decide where to start any job and be able to develop a plan for completing that job with a minimum of supervision. They should also be able to determine when problems arise which block progress and immediately get help from the systems analyst who originally defined the job.

It is very helpful if the programmer is imaginative and creative as long as these characteristics are exercised within the constraints of logic and the needs of the job at hand. A logical and objective mind is the programmer's most essential characteristic.

*Working conditions.* Often the computer programmer will work as a part of a team with one or two analysts and several other programmers. He or she will most often report to the systems analyst or the programming supervisor in charge of the project. The computer programmer will also work closely with the data preparation clerks, who prepare the programs and data according to the programmer's directions, and with the computer operator, who follows the programmer's instructions when running the jobs. A spirit of cooperation among staff members in a computer center is important for job satisfaction and efficient production.

Computer programmers are most often provided with well-lighted, air-conditioned offices somewhere near the computer room. They will generally have as much quiet and privacy as possible, as their work does require much detailed concentration.

Computer programmers may sometimes be required to work long or unusual hours. This flexibility in working hours is necessary because they may need access to computer time which is often only available during non-production hours.

Programmers frequently have a hand in determining completion dates. As a result, they can be held accountable when the dates are not met, which creates considerable deadline pressure. Since computer time is expensive, the programmer should also realize the need to keep the cost of developing programs within the estimates agreed upon for a job. There is considerable satisfaction for programmers who know they have completed a task on time and with efficient use of computer resources.

Some frustration may develop for programmers who wish to use their skill in creating very "sophisticated" programs. Often the most "sophisticated" program may be the least usable to the customer because it may require more computer time and cost than the customer can afford to spend.

In spite of the pressure of cost limitations and deadlines, the computer programmer can find ways to create new and helpful processes for the computer. If aware of the needs of those who require such skills, he or she can become a much sought-after professional.

## Computer programming as a career

Ever since the first electronic digital computer was developed, there have been fewer qualified computer programmers available than needed. Although the demand fluctuates with the economic cycles, it appears that the shortage will continue. Thus, an experienced programmer can usually find several positions to choose from, which provides the opportunity to choose the city he or she wishes to live in and sometimes even the company for which he or she will work. Very few programming jobs will be found in rural areas or small towns, since most computer installations are found in metropolitan areas.

Programmers may gain experience in such businesses as banking, insurance companies, airlines, or utility companies. In fact, nearly every kind of firm is rapidly becoming interested in computers to solve their business problems. Scientific programmers may find jobs in electronic and defense firms, federally funded projects, or with software firms.[*] Applicants may also take positions in a service bureau which often serves a great variety of business concerns.

Computer programmers' careers can take several paths. They may choose to remain in applications programming, where new developments in computers allow them to refine their skills. In large computer centers, computer programmers can become programming supervisors, systems programmers, or systems analysts. If they have management abilities, they can become computer center managers.

The chances for advancement are excellent for skilled computer programmers. Their effective communication with computers provides them an excellent basis from which to expand their professional abilities.

[*] Software firms are companies which specialize in the development of software.

## THE WORLD OF THE SYSTEMS PROGRAMMER

### Introduction

While the applications programmer makes use of software to aid in creating programs, he or she usually has little interest in the design of the software. The applications programmer is interested only in solving the user's problem and depends on the software to reduce the complexity of the applications programming tasks as much as possible.

The systems programmer, on the other hand, is the person who actually designs the software which makes it possible for the hardware to function most efficiently in processing programs. The systems programmer is probably at least as important a person as the computer designer. This is because a computer system is defined as the computer hardware *along with* the software which keeps it running smoothly.

Usually in the past, a customer would buy or lease a computer and then figure out how to apply it to a problem. But the realization that software may be the most important part of a computer system has led to increased demand for good software to accompany the computer hardware. Often a particular machine may be selected because of its software.

The manufacturers of computers are usually responsible for creating the software that comes with the machine. This means that every manufacturer needs to have a staff of competent systems programmers to design and develop a complete "software package" to go with every computer that is produced. Usually, at the same time that the hardware engineers are building and testing a new line of computers, the systems programmers are working on the software to go with that particular hardware.

As you can imagine, the systems programmer who works for a manufacturer is interested in designing software which will meet the requirements of as many potential users as possible. He or she does not try to develop software uniquely suited to solving a particular problem for a particular customer. This means, then, that the customers (or "users") who have special needs for their computer systems usually hire their own systems programmers. The two types of systems programmers, the manufacturer's and the user's, thus function differently in one aspect. The first *develops* the software and the other *uses* and/or adapts it. Let's take a little closer look at how the jobs of manufacturer's and user's systems programmers differ.

## The manufacturer's systems programmer

When a computer manufacturer has a design for a new computer, the engineer gives the design to the systems programmer and says, in effect, "Here's a new computer. Design the software for it." The engineer is responsible for providing the systems programmer with complete specifications, including the logic the machine uses and its particular "machine language"—the numeric code used to program it. The systems programmer then uses the machine language to create the operating system, language translators, and other kinds of software needed to make the new hardware function efficiently and flexibly.

## The user's systems programmer

When the software has been completed by the manufacturer's systems programmer and the hardware has been completely checked out, the new computer system is put on sale. At this point the interested customers (users) usually call in their own systems programmers. It is part of the job of the user's systems programmer to help in the selection of a computer.

The new computer system normally comes equipped with a large selection of software modules or programs. Once a system is selected, the user's systems programmer must decide which of these modules to order. He or she will then be responsible for seeing that the software is properly installed and that it does what the user wants it to do. To be certain the users are equipped to make best use of a new system, manufacturers always provide the users' systems programmers with training in the new system and its software. This training is then passed on to the programmers. When software modifications are required, the user's systems programmer either makes the changes or arranges with the manufacturer to have it done.

The users' systems programmers are not involved in designing or developing software. Instead they are responsible for installing and adapting the software provided by the manufacturer.

## Requirements for the job

Whether a systems programmer works for a manufacturer or a user, the required skills, education, and personal characteristics are essentially the same. The manufacturer's systems programmer will spend more time in applying skills and training to the designing of software, but the user's systems programmer must be able to understand that design thoroughly, and be able to modify and adapt it to her or his own needs.

*Skills.* A systems programmer must have all of the same skills as an applications programmer. That is, he or she must be able to write programs in high-level languages such as FORTRAN, COBOL, or BASIC. But this is not enough. The systems programmer also must be skilled at programming in assembly language and even machine language, as most of the work will be done in one of these languages.

Because a systems programmer works so closely with a particular machine, particular knowledge and skill in the detailed inner workings of that particular machine will be needed. The systems programmer will need to be almost as much of an expert on that machine as the person who designed the hardware.

Communications skills, both written and verbal, are essential to systems programmers. If they work for a manufacturer, they will need to correctly interpret the software needs for a computer, as communicated to them by managers and marketing personnel, so that the software they design will be wanted, needed, and usable by the largest number of customers. They must also communicate directly with the hardware designers and builders. And they will probably be working as a part of a team of systems programmers, each of whom is responsible for a specific portion of the software, and must communicate frequently with these team members.

If systems programmers work for a user, they will need to provide clear communications to the manufacturer about the software needs of their organizations. As modifications or special purpose software are required, the systems programmers must communicate with the manufacturer either for permission to make changes or to get the manufacturer to do the job for them.

They must also be able to communicate effectively with other programmers (both systems and applications) in their organization, as well as with the systems analysts and their supervisors. In addition, since systems programmers are usually responsible for training the other programmers in using the system and for helping to write users' manuals for the system, they need good skills in clear and logical presentation of ideas in speech and writing.

*Education.* The education required of a systems programmer is usually a college degree. The most useful major fields are computer science, mathematics, engineering, physics, and electronics. In particular, programming should be studied with emphasis on machine and assembly language. A thorough knowledge of how a particular computer works is very helpful.

If you choose to become a systems programmer, the most desirable courses to take in high school are advanced mathematics and English, in that order.

Although most systems programmers receive their training in colleges, there are also manufacturer's classes which offer training in the field of systems programming.

*Personal characteristics.* Programmers are logical thinkers and are thorough and detailed in their work. They get along well with people. They are emotionally stable and can work under pressure. Most programmers are dedicated to their work and willing to work overtime to complete their jobs.

The systems programmer is a special kind of programmer who is more concerned with how the computer does its job than with what it does. The systems programmer should possess the same personal characteristics as the applications programmer but will probably be more curious and creative, less communicative, and more solitary than the applications programmer. The systems programmer welcomes a challenge, and will be dissatisfied with anything he or she is given to use until it has been improved to his or her own satisfaction.

*Working conditions.* Whether working for a manufacturer or a user, the systems programmer usually will be provided with a well-lighted, air-conditioned office. Because much of the work requires concentration and attention to fine detail, his or her employer will probably provide as much quiet and privacy as possible.

Even more than other data processors, the systems programmer frequently works nights in order to have unlimited access to the computer during non-production hours. There is usually a great deal of freedom in working hours, with frequent overtime.

The user's systems programmer will probably report directly to the Computer Center Manager, providing freedom in working closely with both programming personnel and operations personnel. The user's systems programmer will also work in close contact with representatives of the computer manufacturer.

On the other hand, the manufacturer's systems programmer will probably report to an executive who is responsible for development of software systems. He or she will work closely with the designers of the hardware and with other systems programmers who may be developing other porti ns of the same system. The software design specifications will be drawn from a market survey conducted by the manufacturer. They will be based on what users say they want and need. Consequently, systems programmers usually find that their creativity and imagination must be tempered by the commercial requirements of the system they are working on.

The frustrations systems programmers are most likely to encounter will probably result from their employers' insistence on the marketability and/or usability of software. Often, systems programmers would prefer to exercise their ingenuity and skill in creating truly "elegant" software, far more elegant than that specified by their employers—but perhaps not suited to the user's needs, regardless of its elegance. They may be restricted by the need to produce software within certain time and cost specifications. Thus they may not be able to add all of the refinements or "bells and whistles" they might personally prefer.

In spite of such frustrations, however, systems programmers will find many opportunities to create new and useful software in response to demand from users. As long as they remain sensitive to and aware of the software needs of those who will use what they produce, they can easily distinguish themselves as skilled and inventive professionals.

## Systems programming as a career

Most job openings for manufacturer's systems programmers will be found in or near the large cities where the manufacturers are located. User's systems programmers will also be in demand in metropolitan areas, although they can find jobs in smaller cities and university towns as well.

One other potential employer of systems programmers are the "software houses." These companies specialize in producing software which often is better in some ways than that provided by the computer manufacturers. Usually, they are located near large metropolitan areas, where manufacturers and users are concentrated. The work in a software house would be similar to that for a manufacturer.

University computer centers often employ one or more users' systems programmers. The work they do may be supported by funds from a research grant. The work may require the development of new software systems similar to the systems developed by manufacturers. This kind of work may be less user-oriented and more experimental than is ordinarily the case.

Systems programmers usually do not have many career paths open to them. Frequently, their skills and temperament do not fit them for management. The more usual career path is from user's systems programmer to manufacturer's systems programmer—or perhaps to a software house.

If temperamentally suited, the systems programmer may move into management through a programming supervisor's job or an operations management position.

## THE WORLD OF THE SYSTEMS ANALYST

### Introduction

As you move along the line of jobs in a computer center from data clerk and computer operator to computer center manager, you find that the jobs become less routine and automatic and more conceptual. That is, they require more thinking and less repetitive "doing." In this picture, the systems analyst is quite far along the line. In fact, in most computer centers, the only jobs beyond the systems analysts are those in computer center management.

In large computer centers the work of systems analysis may be divided among senior and junior systems analysts. The more highly qualified and experienced people hold the senior positions. The junior post may be held either by a newly trained systems analyst gathering on-the-job experience or by a programmer training to become a fully qualified systems analyst. In the average small computer center, however, one or two equally qualified systems analysts assume the responsibilities for all facets of the analysis work.

But, what is this "work"? What "systems" are analyzed and why? In general, we can say the systems analyst is the master problem-solver of the computer center. While the programmer works out the details of individual programs and the computer center manager coordinates the running of the center's various parts, the systems analyst is busy analyzing general problem situations in the organization or elsewhere and developing special systems of data processing to solve them.

Whether the problem area exists within the organization or in an outside setting, it is frequently presented to the analyst in very vague terms. It is up to the systems analyst, then, to make a thorough analysis of the situation and to define the problems and the needs accurately and completely. He or she must master-mind the solution—that is, analyze all potentials for solving the problems with data processing techniques and create the master-plan for developing the data processing system that would best solve the problems and meet the needs of the situation. Finally, the systems analyst acts as leader of the systems development team consisting of analysts and programmers who will implement the master plan.

Typical problems systems analysts might be expected to solve are those involving payroll systems, accounting systems, airline reservation systems, inventory and ordering systems, statistical and research analyses, attendance and grade reporting or scheduling systems for schools, and so forth.

## Requirements for the job

*Skills.* It is easy to see that if you would rather solve problems on a broad scale than perform specific, well-defined tasks, you might be a good candidate for a systems analyst career. Along with a preference for broad problems, then, what specific skills and abilities must systems analysts have to accomplish all that they do?

When the systems analyst first comes to grips with a general problem, skill at communicating (both in speech and writing) is immediately called upon. It is important throughout the process of solving the problem. First, the systems analyst must be able to communicate effectively with those presenting the problems for solution. In particular, the need to gather as much information as possible about the problems and needs in the situation often requires the systems analyst to be especially skillful at asking questions and interpreting answers.

The analyst must be equally skilled in handling technical communications (e.g., flowcharts, block diagrams, computer language). This is so that (1) plans and ideas can be communicated effectively to the systems development team and (2) the technical aspects of the system can be translated into everyday language for non-technicians when necessary. In addition, since the system analyst's job involves considerable time working with colleagues in the organization and with outsiders (manufacturers, salespersons, and the public), the ability to communicate well with others is vital.

As soon as the systems analyst settles down to planning a needed data processing system, most or all of the following may be necessary: determine data gathering techniques to be used, design forms, select data processing equipment, define specifications for the computer programs to be written, prepare documentation for the system, prepare and conduct training programs for the users of the system, and even help determine the manner in which the output of the system will be used. These tasks require a wide range of knowledge about the organization the system is being designed for. They also require familiarity with data processing equipment and procedures, and their capabilities, limitations, and potentials. A thorough understanding of the programming that supports the computer system is also necessary. Without this range of technical expertise, the systems analyst could not adequately plan systems development, supervise the analysts and programmers on the development team, or communicate the system's use and potentials to the people in need of it.

In addition to these technical realities, the systems analyst must also deal with time schedules, cost estimates, budgets, personnel, and recommendations about organizational structure. Therefore, a general knowledge of how organizations function is highly desirable.

In contrast with programmers and technicians, the systems analyst must operate in a broader world which includes others inside and outside the organization. This necessitates the ability to work with people in all contexts.

*Education.* The skills and knowledge required by the job suggest the importance of college-level education. There is disagreement as to whether a college degree is necessary for a systems analyst. It depends on the nature and size of the organization and the type of work being done. In an educational or research setting where statistics or scientific routines are apt to assume a major role, a college degree is more important than elsewhere. In order to supervise programmers, some of whom might have college degrees themselves, it would be appropriate.

It is safe to say that a man or woman planning a career in the field and having hopes of becoming a respected systems analyst in a significant computer center complex should plan on obtaining at least a bachelor's degree. A master's degree is desirable. Appropriate areas of study are business administration, accounting, statistics, economics, engineering, applied sciences, and liberal arts. Growth and change are so rapid in data processing that the analyst will find it necessary to participate in frequent on-the-job and off-hour training courses to stay abreast of the developments in the field.

*Personal characteristics.* Now, what are the personal characteristics essential to systems analysts? To carry out the actual analysis and development of data processing systems, a strong capacity for logic and analysis is necessary, as well as a natural flair for imaginative and creative problem-solving. Analysts also need the qualities of objectivity and detachment to work with problems outside the setting in which they exist and to strike a balance between the creativity and the practicality of solutions. In addition, systems analysts must have the natural breadth of vision that will allow them to see the "big picture" while looking at a segment of it.

Because systems analysts are responsible for scheduling work, organizing future tasks, and coordinating the work of others, they need to be orderly and well-organized. Initiative should be natural to them for much of the work requires both self-motivation and the ability to forge ahead seeking solutions to problems never solved before.

Dealing as they do with people at all levels of authority and responsibility, systems analysts should be tactful, diplomatic, and professional in their manner. As we saw before, the better their powers of communication and persuasion, the more effective they will be in their position, where so much interaction with people is necessary. Since an essential ingredient in communication is the ability to put oneself in the place of others and to appreciate other points of view, empathy is a most valuable characteristic for the systems analyst to have.

*Working conditions.* Whatever the size of the computer center, the systems analyst usually finds herself or himself in a well-lighted, air-conditioned office with new equipment and furnishings. Not far away are the programmers' offices or work area. The analyst may expect to be alone working in this modern office only part of the time, however, since the job calls for frequent contacts with management, administrators, clients, the development team, and others. Because of frequent need to communicate, secretarial assistance, reproducing equipment, and other office services will usually be available. Like the other workers in the center, an analyst normally works regular hours, but this schedule may be subject to irregularities depending on availability of computer time.

The systems analyst often works under the combined pressures of time and budget. Frustrations are not infrequent in the work, as pet projects fail or compromises must be made. In addition, it isn't entirely uncommon for the analyst to find the system to be the target of criticism or animosity by those who are reluctant to accept or understand change.

On the positive side, however, the systems analyst has frequent occasion to enjoy the sense of pride and accomplishment that results from a job completed successfully. In addition, as a part of the data processing team, the systems analyst will experience a vital group unity that comes from being part of a new, dynamic, and productive profession.

### Systems analysis as a career

The demand for systems analysts fluctuates with the economic cycles, but the general trend is toward an increasing shortage of qualified candidates for the positions available. Experienced analysts are particularly in demand.

The majority of systems analyst jobs are found in the metropolitan areas where the large corporations are located. Here, many analysts work for consulting firms which provide their services to companies that either have no staff of analysts or whose analysts cannot solve a particular problem. Systems analysts can also find jobs on the data processing staffs of large industries and corporations, banks, hospitals, insurance companies, educational institutions (universities, school systems, etc.), all levels of government, special projects (NASA, for example), and others. In smaller installations, the functions of programmer and analyst are frequently combined in one job.

The career path of the systems analyst can take any number of interesting directions, depending on the person's talents and ambitions. First, he or she may become a senior systems analyst and continue to expand experience and expertise in the specialties of the job. From here the next step may be to become an acknowledged top-flight specialist in a field. Instead the next step may be to move in the direction of management, becoming a systems supervisor, director of systems and programming, or computer center director. If talents lie in the direction of administration, the capable systems analyst may advance into an administrative assistant's post and other administrative positions.

## THE WORLD OF THE COMPUTER CENTER MANAGER

### Introduction

Anytime a group works together, organization and leadership must exist to ensure smooth and efficient operation. A computer center is no different. In small computer centers, where the staff is not large, one person frequently assumes all of the management tasks. In larger centers, where there are many employees doing a large volume of work, the leadership necessary to organize and coordinate the activities of the staff comes from a group of management jobs.

### The jobs in computer center management

Computer center managers hold the top jobs in the computer center. Their jobs also mean top responsibilities. They are responsible for directing and coordinating the efforts of all the other people on the computer center staff.

The chart in Fig. 7-3 will give you an idea of the organization of a typical computer center and the relationship among the managers.

Computer center management usually includes the three positions shown on the chart:

- *The Operations Manager,* who is responsible for all machine operations and supporting services.
- *The Systems and Programming Manager,* who is responsible for all activities connected with the design, programming, and documentation of data processing systems.

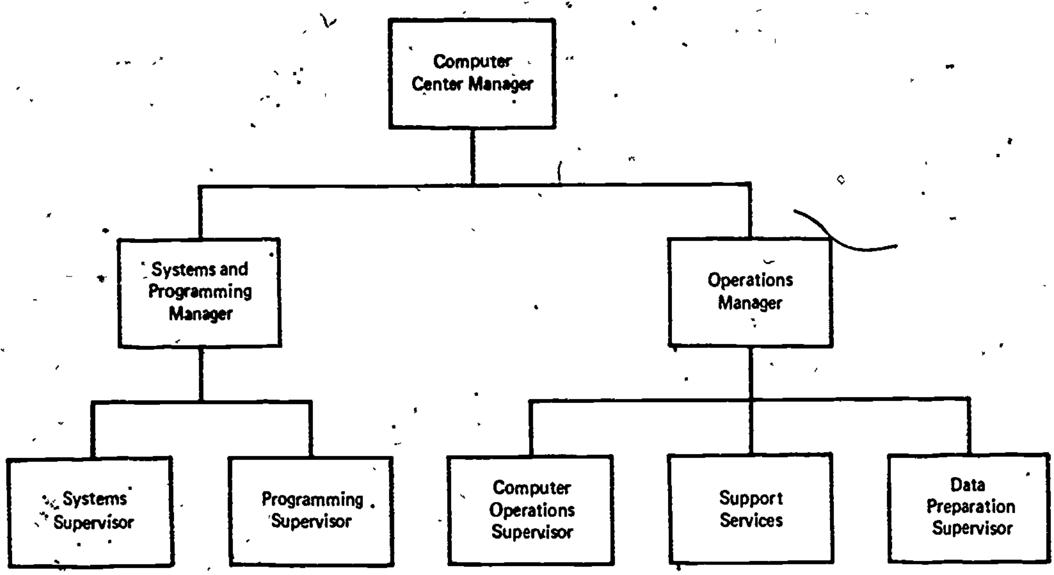


**Fig. 7–3** Organization of management in a typical computer center

- *The Computer Center Manager,* who is responsible for the overall direction and control of the computer center.

A large or rapidly growing center might also have a Business Manager and/or a Hardware Development Manager. A service bureau, which sells computer services, would usually have a Marketing Manager.

***The operations manager.*** The Operations Manager is responsible for the smooth running and productivity of the center's Operations Division. This includes responsibility for the operation and output of the computer hardware, all data preparation personnel and equipment, and the support services which "back up" the operation, such as the tape library and the stock room supplies.

Of all the managers in a computer center, the Operations Manager is the least likely to be found at his or her desk, because responsibilities for the machine operations require considerable time in the computer room or the data preparation room overseeing the work and checking to see that operations are on schedule and deadlines are being met.

In larger centers the manager may have one or more supervisors to assist in the supervision and evaluation of work and personnel. In smaller centers, the Operations Manager would handle this supervision, including orienting new employees and teaching them their jobs. In these instances, the Operations Manager usually has enough knowledge of the machines to overcome the problem. In the case of major problems, he or she is usually responsible for deciding whether to call the maintenance personnel.

The Operations Manager is responsible for the scheduling of all the equipment and hardware in the division. There is, consequently, frequent occasion to work closely with systems analysts in preparing production schedules. Since a part of keeping on schedule is to maintain the machines in good working order, the manager always sees that routine maintenance schedules are kept.

The efficient control and distribution of the computer output are a central concern of these managers, and they are responsible for handling any complaints from customers about late delivery of printouts, wrong number of copies being delivered, and so forth. In line with this part of the job, they must see that a check is frequently made of the storeroom to be sure that there is always an adequate supply of all paper, cards, and forms on hand.

They are also responsible for the accuracy and promptness of customer billings, so they must see that all orders and records are accurately kept in the division. In addition, since the records and materials used are valuable, they must be certain that the security of the operations and data storage areas is maintained.

As managers of a specialized division, it is up to them to select and provide for the training of personnel working in the division. This part of the job involves not only filling current vacancies in the operations staff, but preparing long-range forecasts of the division's work load to estimate the future personnel needs.

The long-range forecasts an Operations Manager must develop are also essential to preparation of the division's budgets, which is a basic part of the job as manager. The budgets for the division anticipate the income and expenditures of resources for the division for a certain period.

***The systems and programming manager.*** The Systems and Programming Manager is responsible for the systems development and programming groups which design, prepare, implement, and maintain the data processing systems. In contrast to the Operations Manager, the Systems and Programming Manager is apt to spend considerable time at his or her desk dealing with the people and paper work that are central to this job.

New systems analysts and programmers on the staff are all selected and usually trained by this manager. Large centers may also have supervisors who may assist with the orientation and training of new employees. The work schedules, priorities, and specific job assignments for the systems and programming staff always originate with the manager. In addition, the manager keeps himself or herself available to the staff to help when difficult problems are encountered in the work.

Along with maintaining effective communications with the entire staff, the Systems and Programming Manager is responsible for the center's contact with clients outside the center. It is a vital part of the job to communicate with customers, interpret their requests for computer services, help them to understand what the computer can do for them, and to direct them to a systems analyst for the detailed planning of the computer solution to their problem. The manager usually helps create acceptance standards with the customer for whatever program or system is required. Once a program or system has been completed for a customer, the manager usually takes the major responsibility in contacting the customer about it and in maintaining contact with the customer to be certain it runs properly. When an entire system has been developed, the manager ordinarily participates in the "buy-off"—that is, the process of the user's accepting the complete system.

Like the Operations Manager, the Systems and Programming Manager is responsible for making long-range forecasts and plans for personnel and facilities for the division. On the basis of these forecasts, she or he creates the budget for the division and sees to it that projects are funded and that they are finished on schedule.

It is important for the Systems and Programming Manager to stay abreast of the new developments in the data processing field. In addition, the Systems and Programming Manager usually keeps in contact with other managers to keep them informed of available computer services, and to find out what new services may be needed that the staff could develop.

***Computer center manager.*** The Computer Center Manager (called the Director of Data Processing in some firms) is ultimately responsible for all the activities of the computer center. In small centers where he or she may be the only manager, the Computer Center Manager must assume the jobs of overseeing both the operations work and the systems and programming work as well as the tasks of the overall center management. In larger centers, there are usually division managers assigned to oversee the divisions' work while the Computer Center Manager focuses attention on providing leadership and organization for the center as a whole.

As the top manager, the Center Manager is basically concerned with organization and communication. In general, he or she is responsible for establishing and communicating the center's broad objectives to the other managers and staff. It is also up to the Center Manager to forecast, plan, and budget for the center as a whole. This part of the work relies heavily on the forecasts and reports from other managers, so establishing clear channels of communication with them is important.

Any new division or department managers added to the center's staff are selected and trained by the Center Manager. This person is also responsible for coordinating the work of all the managers on the staff.

It is usually up to the Center Manager to make all final decisions on the center's expenditures or personnel. But these decisions are likely to be based on the recommendations of other managers. If personnel conflicts develop within the center, the Center Manager may be called on to help settle the issue.

This manager is the person from the center who is responsible for dealing with higher level management in the company. Frequently the Center Manager meets with the boss, perhaps a vice-president of the company, to discuss overall policy of the center, budget plans or problems, facility needs, and so forth. He or she is generally responsible for representing the center as a whole and for explaining the function of the center to the company's top management (president, vice-president, and board of directors).

### Requirements for the job

***Personal skills.*** Some people picture a manager as a person who stands threateningly over his or her employees, scowling and urging them on, making them do their jobs through fear or force. You would probably agree that it would be terrible to work for such a person, and it would be terrible to be such a manager.

Fortunately, this concept of a "slave-driver" manager is a false one. Since people desire to do a good job when the job is important and not distasteful, the manager does not need to "drive" the employees. Instead, the manager sees to it that they have all the tools and facilities they need to do their job well, and that they understand what the job is and why it needs to be done.

To help the staff do its jobs well, it is desirable for the managers in a computer center to have backgrounds in all phases of computer work. They need an understanding of the fundamentals of systems design, programming and computer operation, information storage and retrieval techniques, data analysis, and hardware and software evaluation.

But the managers' jobs require something in addition to a knowledge of tasks and procedures in data processing. They require skills in communication, leadership, organization, and human relations, for the managers deal not just with jobs but with people.

Like managers in any business, the managers in computer centers must be able to forecast personnel and facility needs, which involves skills in estimating, planning, and budgeting. Frequently, they will be called on to put such information in writing, or to write progress or evaluation reports. Obviously, a well-developed ability to communicate both in speaking and writing is important for work in management positions.

Whether the individual is the Computer Center Manager or a manager of a specialized division in the center, a high degree of personal integrity and a deep loyalty to staff and employer are necessary. Personal traits should mark the Computer Center Manager as a person who is straightforward, fair, tactful, well organized, and imaginative. Good judgment will constantly be called into use.

*Education.* An individual planning a career in management in data processing should have a bachelor's degree. He or she will be supervising the work of other individuals who have such a level of education, and the demands on skills and energies require at least such a background. Further education at the graduate level would prepare anyone even better for the challenges of this job.

Desirable areas of specialization are business administration, accounting, statistics, economics, engineering, applied sciences, or liberal arts. The candidate for a managerial position should have training in management concepts and skills. Once on the job, of course, a broad knowledge of the company with which one is working will be needed. The rapid changes in data processing technology and management methods will make frequent additi nal training necessary.

7

*Working conditions.* Data processing management has no place for the "lone wolf." The group of managers described in this manual works together as a team, each member contributing a specialty. Because of the technical nature of the work and the wide range of its application, it would be almost impossible for one individual to keep track of all that is going on in a computer center.

Computer center managers generally enjoy excellent working conditions in the form of modern office facilities, secretarial assistance, and good library, reproduction, and other services. Like other managers, they are expected to do whatever is necessary to get the job done. Consequently, they frequently find themselves donating weekends and evenings to the cause.

Managers work under the frequent pressures of time and budgets. The part they play in a rapidly changing technology is stimulating, but it can also be frustrating as they realize that they cannot move fast enough to keep up with the changes. They may also experience frustration in gaining approval for their plans and projects. While trying to keep pace with the fast changing technology, they must be prepared to find themselves the targets of unjust criticism by those who consider change threatening, and who resist it.

On the other hand, the management personnel at a computer center enjoy the sense of pride and the feeling of group unity that comes from working together in a new and rapidly growing profession.

## Computer center management as a career

The demand for computer center managers follows a general upward trend in the field. More and more computers are being installed. The need for people to manage their use increases proportionately. Since the majority of computers are found in large cities, most management jobs are naturally found there as well. In smaller installations, two or more of the management positions described may be combined into one managerial job.

There are several routes a person may take to computer center management. He or she may progress from a job in operations to Operations Manager, move from programming or systems analysis to Systems and Programming Manager, or cross over from one area to another.

One of the subordinate positions in the organization, not related to

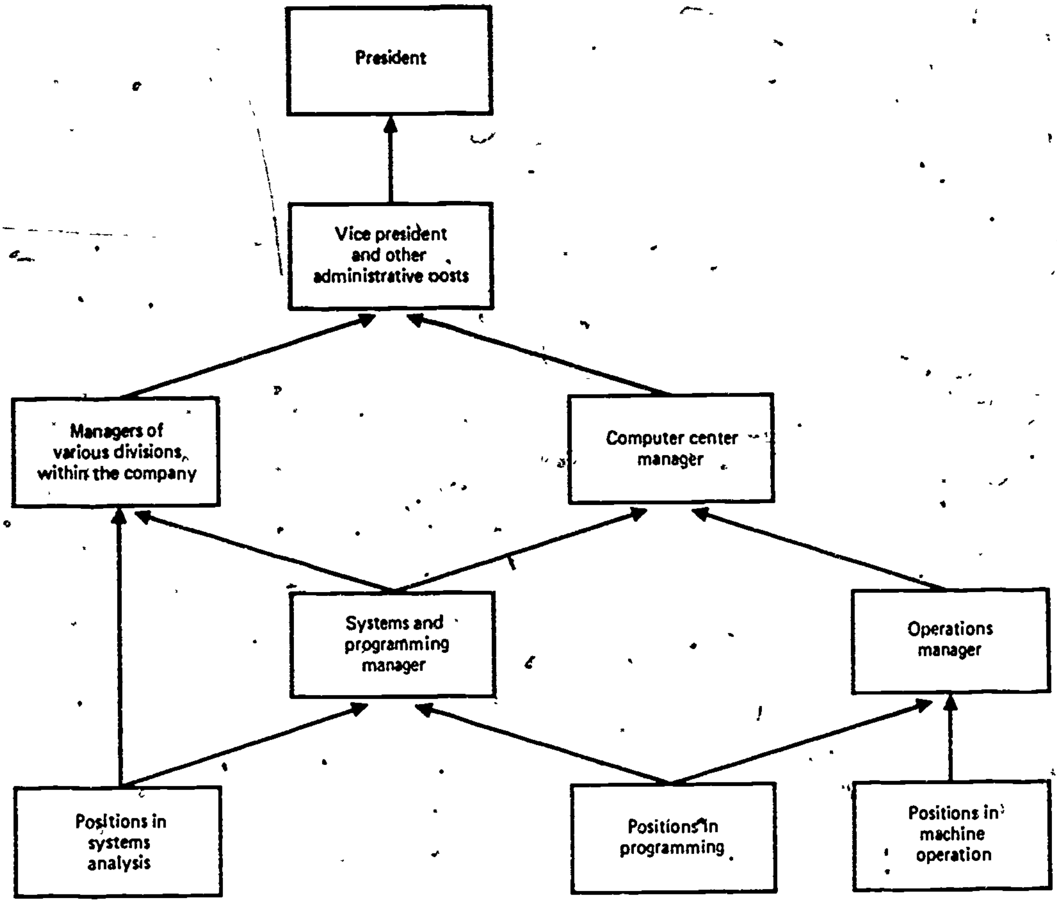

President
Vice president and other administrative oosts
Managers of various divisions within the company
Computer center manager
Systems and programming manager
Operations manager
Positions in systems analysis
Positions in programming
Positions in machine operation

**Fig. 7–4** Career paths in data processing

data processing, could lead to a place as Computer Center Manager. A person could also move from Computer Center Manager to higher management positions in his or her organization, such as comptroller or vice-president. Though data processing management has traditionally been a "dead-end" career, recent trends seem to include data processing people in considerations for high positions such as those just mentioned.

The chart in Fig. 7-4 will give you an idea of the various directions an individual could go in management, whether in shifting from one area to another, moving up within the computer center, or moving into other positions in the organization.

# APPENDIX

## BINARY CODES

The technique of representing letters and numbers inside a computer as combinations of zeros and ones makes it possible for computers to be built entirely from elements which have only two states—for example, *off* or *on*. Although this two-digit number system is cumbersome for humans to use, it is a very efficient, economical, and reliable system for computers. Using it, numbers can be represented by combinations of a pulse or no pulse, a hole or no hole, magnetization in one direction or the opposite direction, a switch open or closed, a light on or off, one or zero, and so on.

A single binary digit, a zero or a one, is called a "bit." A bit is always in one state or the other. It is either one or zero, on or off. It cannot be halfway between.

There are many ways of representing, or *coding*, numbers and letters using only two symbols. If we only needed to code the 10 digits of our decimal number system, 0 through 9, we could use a simple *numeric* code with four bit positions, as shown below:

| *Decimal Digit* | *Binary Code* |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| 8 | 1000 |
| 9 | 1001 |

If we include the 26 letters of the alphabet, plus a few special symbols like + and — and =, we need to use more bit positions, or ones and zeros. Such a code is called an *alphameric* code, since it includes letters of the alphabet as well as numbers. We need to use six ones and zeros to specify each number or letter so that it has its own code. One such six-bit alphameric code is shown below:

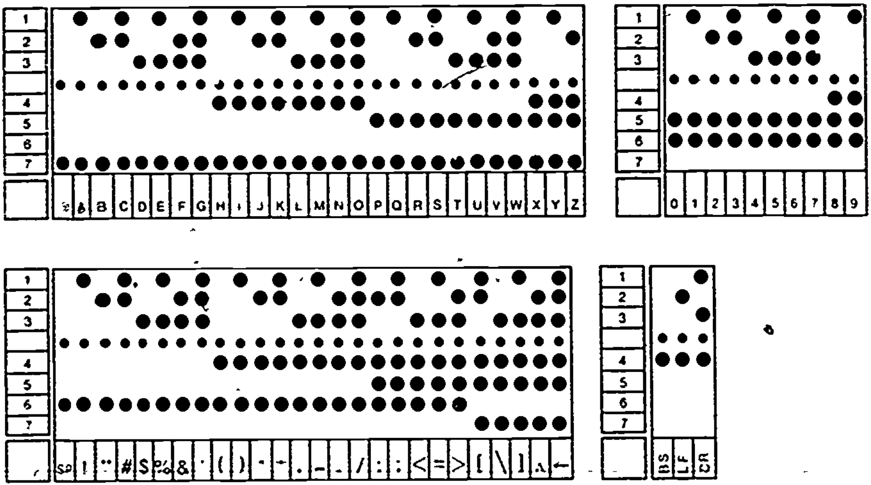
Six-bit ALPHAMERIC Code

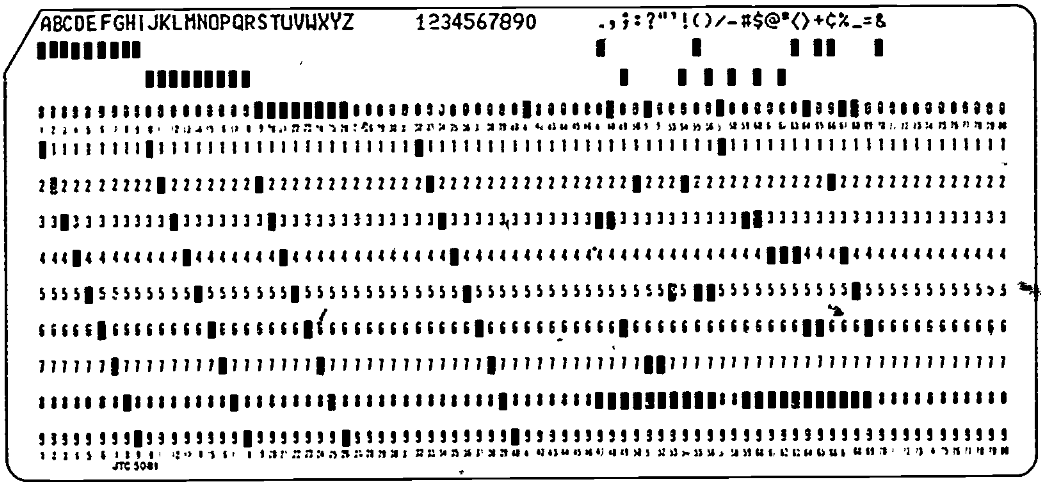
| | | | |
|---|---|---|---|
| 0 | 000000 | ↑ | 100000 |
| 1 | 000001 | J | 100001 |
| 2 | 000010 | K | 100010 |
| 3 | 000011 | L | 100011 |
| 4 | 000100 | M | 100100 |
| 5 | 000101 | N | 100101 |
| 6 | 000110 | O | 100110 |
| 7 | 000111 | P | 100111 |
| 8 | 001000 | Q | 101000 |
| 9 | 001001 | R | 101001 |
| [ | 001010 | — | 101010 |
| # | 001011 | $ | 101011 |
| @ | 001100 | * | 101100 |
| : | 001101 | ) | 101101 |
| > | 001110 | ; | 101110 |
| ? | 001111 | , | 101111 |
| (space) | 010000 | + | 110000 |
| A | 010001 | / | 110001 |
| B | 010010 | S | 110010 |
| C | 010011 | T | 110011 |
| D | 010100 | U | 110100 |
| E | 010101 | V | 110101 |
| F | 010110 | W | 110110 |
| G | 010111 | X | 110111 |
| H | 011000 | Y | 111000 |
| I | 011001 | Z | 111001 |
| & | 011010 | ← | 111010 |
| . | 011011 | , | 111011 |
| ] | 011100 | % | 111100 |
| ( | 011101 | = | 111101 |
| < | 011110 | ." | 111110 |
| \ | 011111 | ! | 111111 |

There are other alphameric codes—for example, the seven-bit ASCII (American Standard Code for Information Interchange) code which is used to transmit data via teletypewriters. ASCII is shown next, followed by a diagram of the code punched on paper tape at the top of page 337. Can you match up characters using the binary code from the table and the punched code on the tape? Do you see how a hole in the tape represents a one in the binary code, and vice versa? Find the letter "V" in binary and then find the "V" punched on the tape.

ASCII Code

| *Symbol* | *Binary Code* | *Symbol* | *Binary Code* |
|---|---|---|---|
| A | 1000001 | 6 | 0110110 |
| B | 1000010 | 7 | 0110111 |
| C | 1000011 | 8 | 0111000 |
| D | 1000100 | 9 | 0111001 |
| E | 1000101 | space | 0100000 |
| F | 1000110 | ! | 0100001 |
| G | 1000111 | " | 0100010 |
| H | 1001000 | # | 0100011 |
| I | 1001001 | $ | 0100100 |
| J | 1001010 | % | 0100101 |
| K | 1001011 | & | 0100110 |
| L | 1001100 | ' | 0100111 |
| M | 1001101 | ( | 0101000 |
| N | 1001110 | ) | 0101001 |
| O | 1001111 | * | 0101010 |
| P | 1010000 | + | 0101011 |
| Q | 1010001 | , | 0101100 |
| R | 1010010 | — | 0101101 |
| S | 1010011 | . | 0101110 |
| T | 1010100 | / | 0101111 |
| U | 1010101 | : | 0111010 |
| V | 1010110 | ; | 0111011 |
| W | 1010111 | < | 0111100 |
| X | 1011000 | = | 0111101 |
| Y | 1011001 | > | 0111110 |
| Z | 1011010 | [ | 1011011 |
| 0 | 0110000 | \ | 1011100 |
| 1 | 0110001 | ] | 1011101 |
| 2 | 0110010 | ∧ | 1011110 |
| 3 | 0110011 | ← | 1011111 |
| 4 | 0110100 | line feed | 0001010 |
| 5 | 0110101 | carriage return | 0001101 |


1
2
3
4
5
6
7
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4 5 6 7 8 9
SP ! " # $ % & ' ( ) * + , - . / : ; < = > [ \ ] ∧ ←
BS
LF
CR



There is also, in common use on magnetic tape, an 8-bit alphameric code called EBCDIC, which is used to represent 256 unique letters, numbers, and special characters.

The Hollerith code, used with punched cards, is another type of binary code. A hole punched in the card stands for a one, and no hole for a zero. The way the Hollerith code works, each column in the card represents one number or letter. In each column there can be no more than three holes punched, and there are 12 possible places for punches in each column. Because of the "no more than three" rule, it takes 12 bits to represent a single character. Examine the Hollerith card above. Which kinds of characters are represented by one hole punched in the column? Which kinds have two holes? Three?

# GLOSSARY

**Applications programs** User-supplied programs to direct the computer in processing particular data to get particular results.

**Arithmetic-logic unit** Part of the central processing unit responsible for symbol manipulation in the processing operation.

**ASCII** *A*merican *S*tandard *C*ode for *I*nformation *I*nterchange; a seven-position punched-hole code used frequently in punching paper tapes.

**Assembler** The simplest kind of translater, which translates from assembly language into machine language.

**Assembly language** A programming language which uses English-like symbols (mnemonic codes or operation codes) to represent particular computer operations, grouped together with numbers indicating addresses to make a complete instruction, e.g., ADD 1202.

**Batch processing** The mode of processing which enters all input into the computer in one batch and may receive all output in one batch sometime later.

**BASIC** *B*eginner *A*ll-purpose *S*ymbolic *I*nstruction *C*ode, an easy-to-learn procedure-oriented language suited to programming many different kinds of problems.

**Basic systems documentation** The set of diagrams, flowcharts, and tables written by the systems programmers which describe the systems software.

**Block diagrams** A diagramming technique used by systems analysts to break a large problem down into its general sections and to show their logical sequences.

**Binary code** A two-state code used in computer processing, e.g., 1-0, clockwise-counterclockwise, pulse-no pulse, etc.

**Buffer** Electronic mechanism in the computer system which holds input (input buffer) and transfers it to the CPU when the processing unit is ready, or which holds output from the CPU (output buffer) until the output devices are ready.

**Card layout** A sheet showing exact positions for items of information on punched cards, often with keypunching instructions.

**CARDIAC** A small simulated computer developed by Bell Telephone Laboratories.

**Card punch** Output device which the computer controls to output information on punched cards.

**Cathode-ray tube** A television-like picture tube on which input and output are displayed; a CRT.

**Central processing unit** The piece of computer equipment responsible for processing all data; the CPU.

**Chain printer** Common impact printer in which hammers press paper forward onto characters on a circular steel chain.

**Character printer** Input/output devices which print one character at a time, e.g., teletypewriter.

**COBOL** *CO*mmon *B*usiness *O*riented *L*anguage; a procedure-oriented language suited to programming business problems.

**Compiled program listing** The printout from the computer which lists each program instruction and identifies errors while the program is being translated (compiled) into machine language and stored.

**Computer** Device which electronically receives data, processes it as directed by programs, and outputs information.

**Computer console** Equipment on CPU used for communicating with the computer, usually includes console typewriter and control panel.

**Computer hardware** The equipment and devices which are used by computer systems in their operations, such as input output devices, terminals, CPU, etc.

**Compiler** The type of translator which translates from English-like programming languages to machine language.

**Complex terminals** Various input and output devices used together as remote terminals.

**Computer programmer** A person who writes applications programs for computers.

**Computer operator** Person responsible for operating and overseeing the computer system.

**Computer system** Complete installation of computer equipment including input devices, CPU, output devices, and secondary storage devices, designed to perform specific data processing jobs.

**Console typewriter** Typewriter on computer console used by operator for communicating with the computer to control its operation.

**Control panel** A set of incandescent lights on the computer console which displays the contents of some internal registers in the CPU; used to monitor the operations going on.

**Control unit** Part of the central processing unit which controls the sequence of processing operations.

**Cores** Doughnut shaped devices in CPU's primary storage which register data in the form of clockwise or counter-clockwise magnetization.

**Data** Raw material to be processed.

**Data clerk** Person who performs job of data preparation, usually a keypunch operator.

**Data file** Any batch of data stored on an input/output medium.

**Data preparation** The step in computer processing in which data is prepared on media in a form which can be input into the computer, e.g., holes on cards.

**Data processing system** A system which processes input data and outputs information.

**Debug** To correct, or take the "bugs" out of, a computer program.

**Decision tables** A diagramming technique used by systems analysts which shows what actions will be taken when any combination of conditions holds.

**Desk checking** The initial debugging step a programmer uses, in which first the program listing is hand-checked at the programmer's desk for clerical and logical errors, and second the compiled programming listing is hand-checked at the desk.

**Device** A piece of equipment which can record codes onto a medium or read code from a medium.

**Direct-entry** An electronic keyboard device which encodes data directly onto a magnetic medium for inputting into computer, e.g., direct-entry key-to-tape or key-to-cassette devices.

**Drive-sprocket holes** Tiny holes running down paper tape media which guide the tape through the punching or reading device.

**Dump routine** Part of diagnostic routine which prints contents of each storage location at any specified checkpoint in a program.

**EBCDIC** *E*xtended *B*inary *C*oded *D*ecimal *I*nterchange *C*ode; code used on magnetic tape medium.

**Electrical pulses** Method by which data is transmitted from a remote terminal or adjacent input device to the electronic computer, over telephone or other wires.

**Feasibility study** In systems analysis, the study of an organization's needs to determine the desirability of using a computer to help accomplish some goals.

**Fields** Sections of punched cards designated for particular pieces of data.

**Flowchart** A step-by-step diagram showing the logical sequence of events in solving a problem.

**Flowcharting template** A plastic plate with flowcharting symbols cut out on it which programmers use when drawing flowcharts.

**FORTRAN** *FOR*mula *TRAN*slator, a procedure-oriented language suited to programming scientific and mathematics problems.

**General documentation** Written information about a program which includes a final flowchart, program listing, program description, and input and output formats (instruction sheets and layout forms).

**Hollerith card** Standard punched card invented in 1890 by Herman Hollerith, data is punched onto Hollerith card in coded holes (Hollerith code) by keypunch machine or card punches and are input by card readers. Cards have 12 rows and 40 or 80 columns for punched holes.

**Hollerith code** Binary code for punched holes representing all letters, digits 0–9, and many symbols, used to encode Hollerith cards.

**Impact printer** Printer which prints using a type bar or wheel pressed against the paper, e.g., chain printer.

**Information system** Computerized system in which data is interrelated and cross-indexed for immediate retrieval and use.

**Input** Information put into a system.

**Input forms** The forms on which data are recorded which are to be processed; forms vary from bookkeeping sheets to specialized forms designed by systems analysts.

**Interactive processing** The "conversational" mode of time-sharing in which user enters input, promptly receives output, and may continue to input and get output as desired.

**Intelligent terminal** A remote terminal with a mini-computer of its own and one or more input and output devices; sometimes a secondary storage device or two are included. It can serve both as a remote terminal for a central terminal elsewhere and as a mini-computer in its own right.

**Job set-up sheet** A part of specific documentation which describes special steps the computer operator must take to run a particular job.

**Keypunch machines** Keyboard machines used to punch Hollerith cards with data in the form of coded holes, to be read into the computer by punched card reader devices.

**Layout sheet** A sheet showing exact positions of items of information on input or output media.

**Light pen** Pen-sized tube with electronic device in it and a light bulb at its tip, used to transmit signal from the location on a CRT screen to which the light pen is pointed by the user.

**Line printer** Output device which prints output from the computer one entire line at a time; it is run electronically by the computer.

**Local mode** *See:* Off-line.

**Machine language** The set of symbols, usually a binary code transcribed in 1's (ones) and 0's (zeros), which the computer itself can understand.

**Macro-flowchart** A flowchart used by systems programmers and systems analysts to show the main steps in the program or system they are working out.

**Magnetic character reader-sorter** Input device which recognizes characters printed in magnetic ink and can sort media so printed or can transform data so printed into electronic pulses the computer can register.

**Magnetic disk** Input/output medium like a phonograph record on which data is encoded in the form of magnetized spots on concentric tracks by a magnetic disk drive; usually kept in packs of 6 or more disks, called disk packs.

**Magnetic drum** Input/output medium in the form of a metal cylinder on which data is encoded in magnetized spots in circular bands by a magnetic drum drive.

**Magnetic-ink character recognition** A technique which uses magnetic-ink in printing characters which can then be recognized by magnetic character reader-sorters.

**Magnetic tape** Input/output medium like recording tape encoded with data in the form of magnetic spots, using EBCDIC code.

**Mark-sense card** Input medium shaped like punched cards but which are encoded by hand using a pencil.

**Medium** A material (e.g., punched card, magnetic tape) on which information is represented in coded form; media are read or encoded by devices.

**Micro-flowchart** Any detailed flowchart showing each step involved in a process.

**Mini-computer** A small central processing unit.

**Mnemonic code** See: Assembly Language.

**Multiprocessor** A central processing unit with more than one arithmetic-logical unit.

**Multiprogramming** A technique in which several programs are in the computer at the same time.

**Non-impact printer** Printer which prints an image by chemical or other non-impact means.

Off-line The operation mode in which a device is not connected to the computer. When off-line, teletypewriters may be used as regular typewriters, if they have an attached paper tape punch, they can also be used off-line to punch tapes.

Off-line storage Secondary storage media such as punched cards stored in a drawer, which must be specially loaded on an input device for use.

On-line The operation mode in which a device can input data into the computer or receive output from the computer.

On-line storage Secondary storage media connected directly to the computer and hence available for immediate use at any time by computer.

Operating system Manufacturer-supplied programs which include special programs, standard routine programs, and translators.

Operation codes *See* Assembly Language.

Operation documentation sheet A part of specific documentation which describes the program(s) and procedures required for running a particular program.

Optical character reader An input device which recognizes specially-shaped characters, or even typed or hand-written characters, and transforms them into electronic inpulses the computer can register.

Optical scanner An input device which inputs data from hand-marked sheets or cards.

Output Information produced as the result of processing.

Output forms The forms in which a data processing system outputs its results, ranging from simple printed reports to special labels and other forms designed by the systems analyst.

Paper tape punch Device which encodes input or output data on paper tape.

Paper tape reader/punch A device which can encode data on paper tape or read data from it, controlled by the computer.

Parity check Eighth hole-position on paper tape medium used to double check accuracy of the data.

Plotter Output device with a movable pen controlled by the computer to draw graphs or designs.

Point of sale terminal Electronic cash register equipped with a scanner for reading magnetically coded information.

Primary storage Storage area in the central processing unit; also called core storage.

Procedure-oriented languages High-level programming languages which resemble programmers own conversational language, e.g., BASIC.

Process Procedure of manipulating data to produce the desired results or output.

Program A set of instructions prepared by programmers, which tell a computer how to process given input, step-by-step.

Program listing The list printed out by the computer of each instruction of a program, used by programmers in debugging programs.

Program loader Utility program in the operating system which supervises the reading in, storing, and starting of new programs loaded into the CPU for use.

Programming language guides Guidebooks provided by computer manufacturers describing how to accurately and efficiently write programs for the particular computer.

Punched card *See* Hollerith Card.

Punched card reader Input device which transforms data from punched hole code into electronic code which can be registered by the computer.

Punched paper tape Paper input medium about one inch wide, any length, which is punched with coded holes by a paper tape punch as computer input or output.

Random access device Any device which can access (locate) data on a storage medium by moving read/write heads directly to desired location, e.g., magnetic disk drive and magnetic drum drive.

Reading wand A pen-like device attached to electronic cash registers which "reads" price and stock information from a magnetic strip on merchandise, credit cards, etc.

Retrieval Obtaining information from computer storage.

Secondary storage Pieces of equipment outside the central processing unit in which quantities of data can be stored.

Serial access Locating data by reading through all data, serially, until desired location is reached, e.g., serial access with magnetic tape.

SKETCHPAD Computer tool for changing rough engineering draft into an accurate design drawing on a display screen.

Software The programs which guide the computer's operations, including the manufacturer's operating system and the user's applications programs.

Source document The original document on which data are recorded which are to be transformed into input form for data processing.

Specific documentation Written information which gives computer operator all the information needed to run a program, including program operation information, job set-up sheet, etc.

Subroutine library The complete set of subroutines provided in a computer's operating system.

Supervisor The portion of the operating system which monitors and controls the operation of other operating systems programs.

System Generally, any set of "elements" which are related to each other so as to achieve a desired goal.

Systems analysis and design The analysis and planning of computer systems installations to fulfill the specific data-processing needs of organizations.

Systems approach The five-step procedure followed by systems analysts in solving systems analysis problems.

Systems flowchart A flowchart of the solution to a systems problem.

Systems software. The programs with which a computer is equipped by the manufacturer.

Teletypewriter Remote input/output typewriter terminal connected to central computer by telephone lines, often with an attached paper tape punch and paper tape reader.

Terminal A device used for sending information to or receiving information from a distant computer system.

Test routine A utility program in the operating system which identifies machine or program errors during operation of programs.

Time sharing Process by which remote terminals may share time on a central computer and input data for immediate processing by the computer.

Tracing A part of desk-checking a program by tracking the overall logic of the program.

Trace routine Part of diagnostic routine which prints the results as each step of a program is executed by the computer.

Translators Programs in the operating system which translate program instructions from programming language into the machine language the computer can work with.

Typewriter terminals Electric input/output terminals which have a typewriter keyboard for data entry and paper-and-type mechanism for printing output; usually appear very much like standard selectric typewriters.

Utility programs Programs in the operating system which perform routine and frequently needed processing jobs such as controlling the transfers of data from one medium to another.

Voice activated checkout system A computerized checkout system which recognizes merchandise names and prices spoken (rather than keyed-in by clerks).

# INDEX